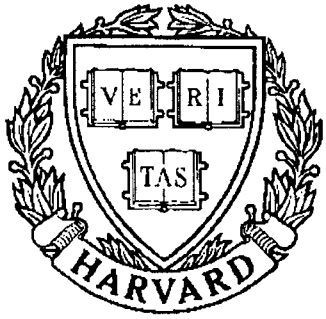
UNDERGRADUATE REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

The Walking Robot Project

by P. Williams, E. Sagraniching, M. Bennett, R. Singh, et al.
Advisor: L. W. Tsai

U.G. 91-4

TABLE OF CONTENTS

Introduction

The objective of the University of Maryland walking robot project was to design, analyze, assemble and test an intelligent, mobile and terrain-adaptive system. This objective was met by thirty engineering students, half electrical and half mechanical, under the instruction of Dr. Lung-Wen Tsai. The project spanned seven months, required approximately 3000 student hours to manufacture, and cost under $7000.

The walking robot project was the subject of two consecutive courses: design in the fall, and construction in the spring. About twenty students participated in each class. The mechanical engineering students divided into two groups, leg and body; the electrical engineering students were divided into the electrical hardware and software groups.

The robot's design became a novel application of existing technologies. The design of the six legs modified and combined well-understood mechanisms and was optimized for performance, flexibility, and simplicity. The body design incorporated two tripods for walking stability and ease of turning. The electrical hardware design employed modularity and distributed processing to drive the many motors. The software design used feedback to coordinate the system and simple keystrokes to give commands.

The construction of the walking machine required precise, distributed work. Mechanical elements were constructed to be functional and durable. Critical machining was performed through numerical control (NC) machinery. Custom circuit boards were constructed and wired to the body of the robot.

The students involved also considered practical factors. The walking machine can be easily adapted to hostile environments such as high radiation zones and alien terrain. Minor modifications would further enable the machine to perform useful tasks with high precision and reliability.

Section I: Leg

The primary goal of the leg design was to create a leg capable of supporting a robot's body and electrical hardware while walking or performing desired tasks, namely those required for planetary exploration. The leg designers intent was to investigate the maximum amount of flexibility and maneuverability achievable by the simplest and lightest leg design. The main constraints for the leg design were leg kinematics, ease of assembly, degrees of freedom, number of motors, overall size, and weight.

Design Constraints

Kinematics was the first constraint considered. It was desired to design a leg with an ovoid walking path to minimize the "slamming" effect caused by a robot's inertial forces during normal walking. This effect is highly pronounced in designs employing a circular kinematic path. The stride length of the leg (the major diameter of the walking path) was an additional kinematic constraint, particularly in designing the leg to climb stairs and maneuver across rough terrain.

The number of degrees of freedom was a constraint pertaining to maximizing the flexibility of the robot while minimizing its weight and complexity. This constraint was closely related to the number of motors used.

The number of motors was limited to thirteen to minimize weight and simplify design. Increasing the number of motors makes a design more flexible but adds complexity to hardware and software design.

Designing for ease of assembly was an important constraint, manifested during the construction of the legs. Six legs were needed in a short period of time; it was therefore important to optimize the design for simplicity while maintaining mechanical integrity.

Modified Four Bar Mechanism

Four bar mechanisms, by definition, consist of a crank link, coupler link, rocker link, and fixed (ground) link. The passive role of the coupler link can be modified by integrating the kinematic paths of the crank and rocker. This is

achieved by replacing the traditional straight bar coupler with an oblique triangular link. The internal angles of the modified coupler can be varied to create an array of continuous, ovoid paths at the disjointed vertex of the triangle. The summation of the crank and rocker lengths must, however, remain less than the fixed length plus the jointed coupler length, in adherence to the rules of kinematics.

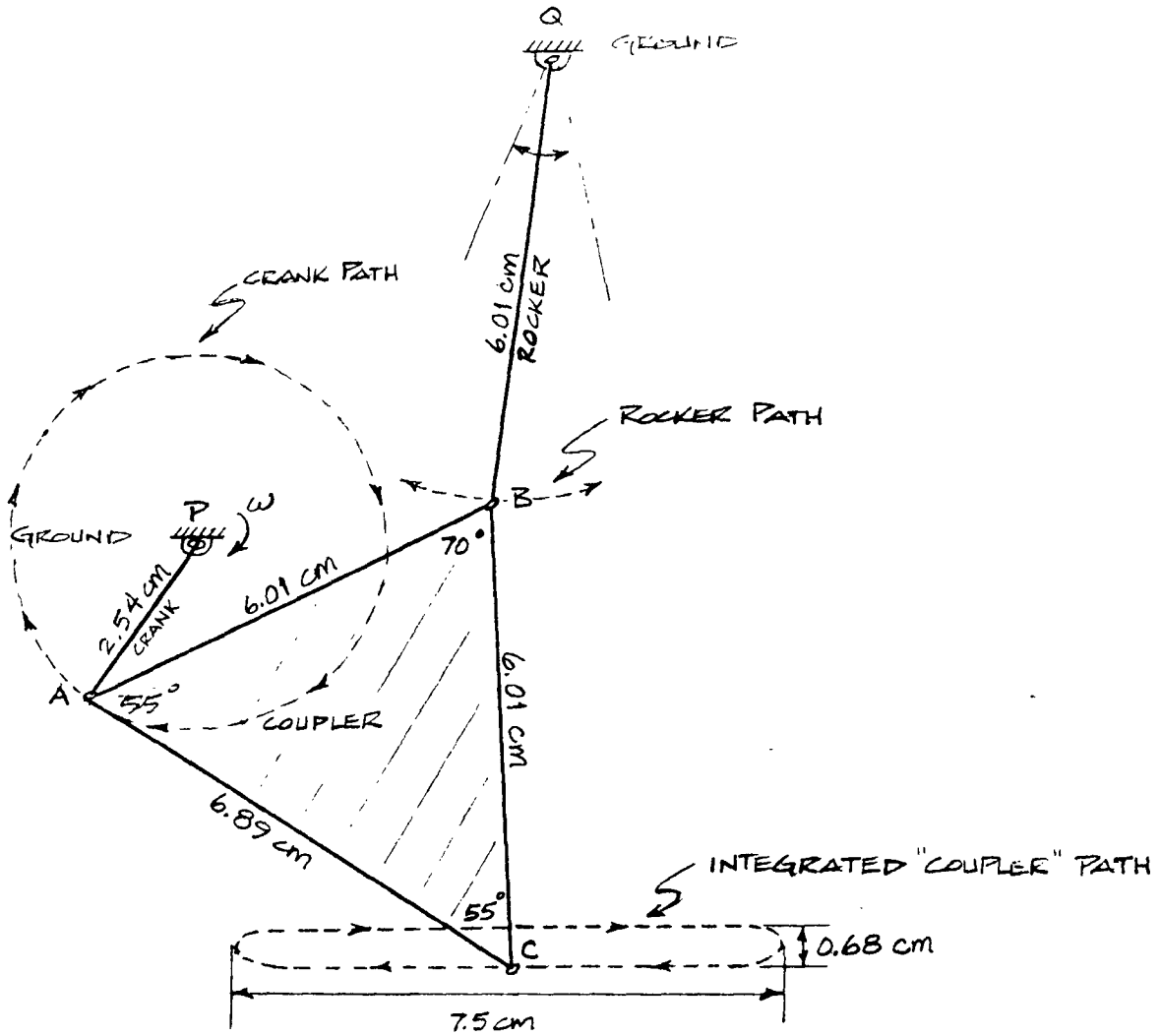
The constraint on kinematics required the leg to have an ovoid path in order to prevent the inertial "slamming" effect during its walking motion. It was also desired that the path be symmetric in order to allow uniform walking motion in forward and reverse. In addition, constraints on power, control simplicity, and the number of motors to be used, required the walking motion to be carried out by one motor. Utilization of the modified four bar mechanism satisfied these constraints and provided the desired walking motion for the robot. Figure 1.1 shows the modified four bar mechanism and the kinematic trace of point "C" through one crank cycle. The major diameter of the ovoid path is 7.5 cm while the minor diameter is 0.68 cm. The internal angles of the triangular coupler link define these diameters and have been manipulated to produce the path that is shown.

The four bar, crank and rocker mechanism (Figure 1.1) is defined by links "AP"(crank), "BQ"(rocker), "ABC"(coupler), and "PQ"(ground). The motor turns the crank through a worm gear combination. As the crank rotates, a pendulum path is created by the rocker link. The crank and rocker links are connected to the modified triangular coupler link, which integrates the kinematic paths of the crank and rocker links, and creates the trace at point "C".

The modified four bar mechanism is an innovative, new mechanism that satisfied the kinematic and motor constraints by providing a smooth, efficient, two dimensional walking motion for the robot through a one degree of freedom system. The path created by this mechanism closely emulates the walking path of humans.

Pantograph Mechanism

The constraint on stride length was 15 cm. This distance was chosen so that the robot can safely maneuver amid small to medium sized obstacles such as rocks and trenches. This was also an appropriate constraint for dynamic and static stability of the robot. Increasing stride length increases vibration and also decreases the robot body's region of stability.



THE MODIFIED FOUR-BAR MECHANISM;
SHOWN TO ACTUAL SCALE.

Figure 1.1

In order to achieve a stride length of 15 cm, the 7.5 cm path produced by the four bar mechanism required amplification by a factor of two. The pantograph mechanism shown in Figure 1.2 is defined by links "CDE"(upper link), "HGE"(lower link), "F'FG"(long link), and "DF"(short link). The pantograph acts as a mechanical amplifier; when attached to point "C" in the position shown, the path created by the crank and rocker mechanism is translated, inverted, and magnified by a factor of two at point "H"(the foot). During normal walking the angle between the lower link and the ground was designed to be 45 degrees at center stride. This maintains a horizontal walking path.

The pantograph was used to magnify the stride length to the desired length of 15 cm, and to provide a means of supporting the robot body and hardware. The combination of the four bar and pantograph mechanisms provides the first degree of freedom for the leg.

The pantograph mechanism acts as a mechanical amplifier for kinematics as well as static forces, therefore, it was important to choose materials and bearings that could withstand these amplified forces. According to kinematic laws, the force at point "F' " (Figure 1.2) is three times that of point "H" and the force at point "C" is two times that of point "H". This demonstrates the importance of considering kinematic constraints in the design.

Leg Lift Mechanism

The ovoid walking path required another degree of freedom in order to avoid obstacles and climb over rugged terrain. The second degree of freedom is in the form of a leg lift mechanism, capable of changing the leg height as well as the stride length. The leg lift mechanism is defined by the pinion gear and lifter gear-link attached to point "F'" in Figure 1.3. The lifter motor turns a worm gear combination which drives the lifter pinion. The lifter gear then rotates, causing the pantograph mechanism to compress or expand, depending on the direction of rotation. The leg was designed to extend *and* compress 7.5 cm from the datum at the foot during normal walking. This results in a total lift range of 15 cm, sufficient to clear small to medium sized obstacles and maneuver within rough landscapes.

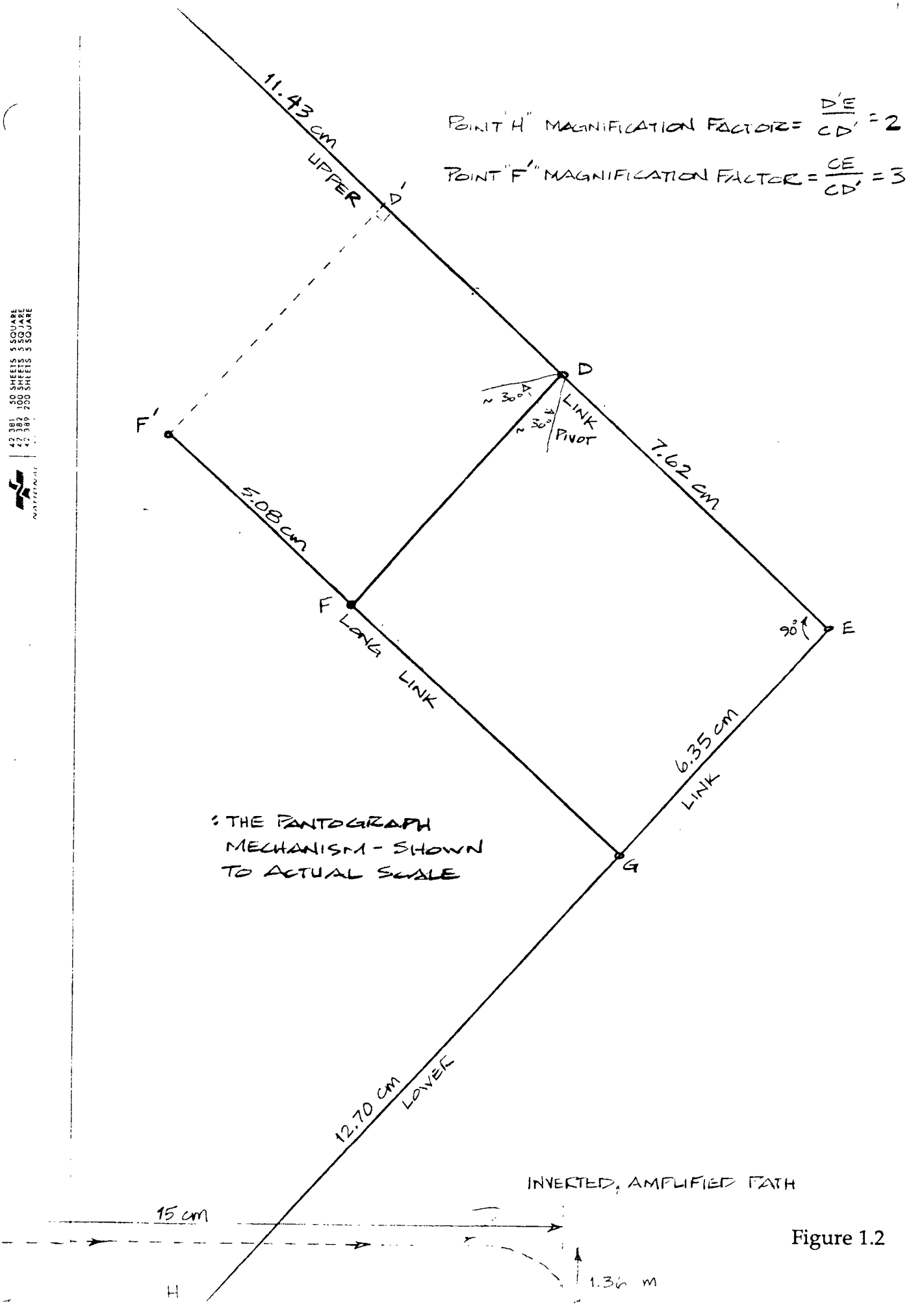

11.43 cm
UPPER
D'
POINT 'H' MAGNIFICATION FACTOR = D'E / CD' = 2
POINT 'F'' MAGNIFICATION FACTOR = CE / CD' = 3
D
~ 30°
~ 30°
LINK
PIVOT
7.62 cm
F'
5.08 cm
F
LONG LINK
90°
E
6.35 cm
LINK
G
* THE PANTOGRAPH MECHANISM - SHOWN TO ACTUAL SCALE
12.70 cm
LOWER
INVERTED, AMPLIFIED PATH
15 cm
1.36 m
H

Figure 1.2

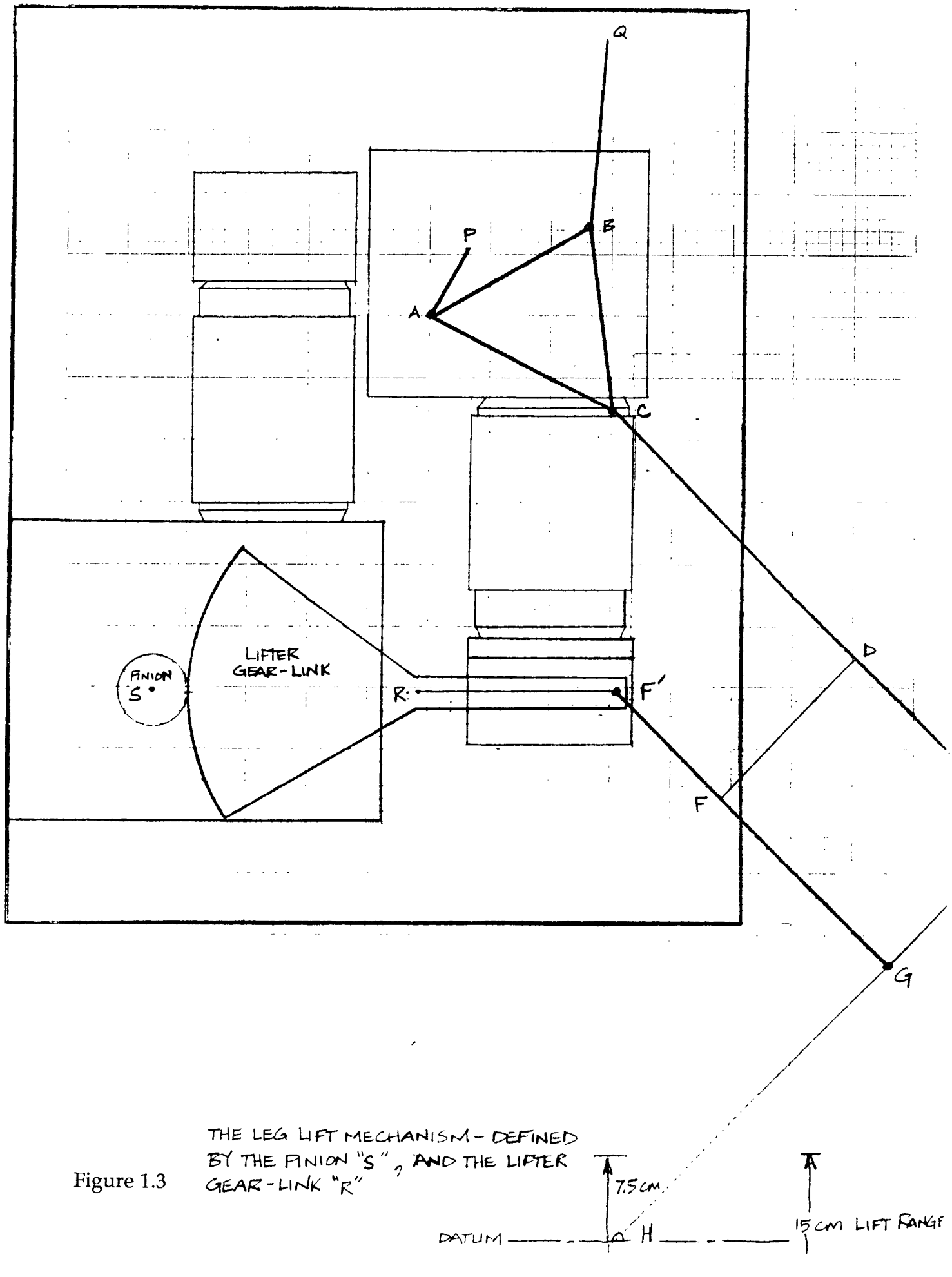


Figure 1.3 THE LEG LIFT MECHANISM - DEFINED BY THE PINION "S", AND THE LIFTER GEAR-LINK "R"

Supporting Structure

The crank and rocker, pantograph, and leg lift mechanisms are supported between two rectangular plates shown in Figure 1.4. These plates provide the ground attachments for the crank and rocker at points "P" and "Q", and also for the lifter mechanism at points "R" and "S". The plates also provide a convenient means for mounting the entire leg assembly to the robot body, and protect the leg links from external objects which could damage or bind the moving links during operation. The motor and gearbox combinations of the lifter and four bar mechanisms are mounted outside the plates to avoid mechanical interference. Motors and gearboxes can be mounted on either side of the two plates, depending on their orientation on the robot body. Three legs have a right hand orientation and the remaining three have a left hand orientation for this design.

The two support plates are rigidly connected by four support columns that are bolted together between the plates. Figure 1.4 shows the entire leg assembly with its top support plate removed.

Mathematical Modeling and Engineering Analysis

The DADS computer software package is a very powerful tool for determining forces, torques, displacements, velocities, and accelerations for a gross array of mechanical elements. The DADS analysis for the final leg design of Figure 1.4 was very involved and was a major aspect of the overall design process. Familiarity with the program took several days. The three main routines within the DADS software are the DADS preprocessor, DADS analysis, and DADS postprocessor.

The preprocessor is used to create the mechanism to be analyzed. This involved creating *system data* (i.e. time intervals, gravitational constants, etc.), *inverse data* (i.e. force coordinates, step size, and tolerances), *revolute joints* (i.e. crank, coupler, rocker, etc.), *body elements* (i.e. ground, crank, and other links), *points of interest* (i.e. points "C", "E", and "H" of Figure 1.4), *driver elements* (i.e. crank and lifter), and *rotational spring-damper elements* (i.e. torsion spring at point "Q").

The analysis routine then uses the information from the preprocessor to

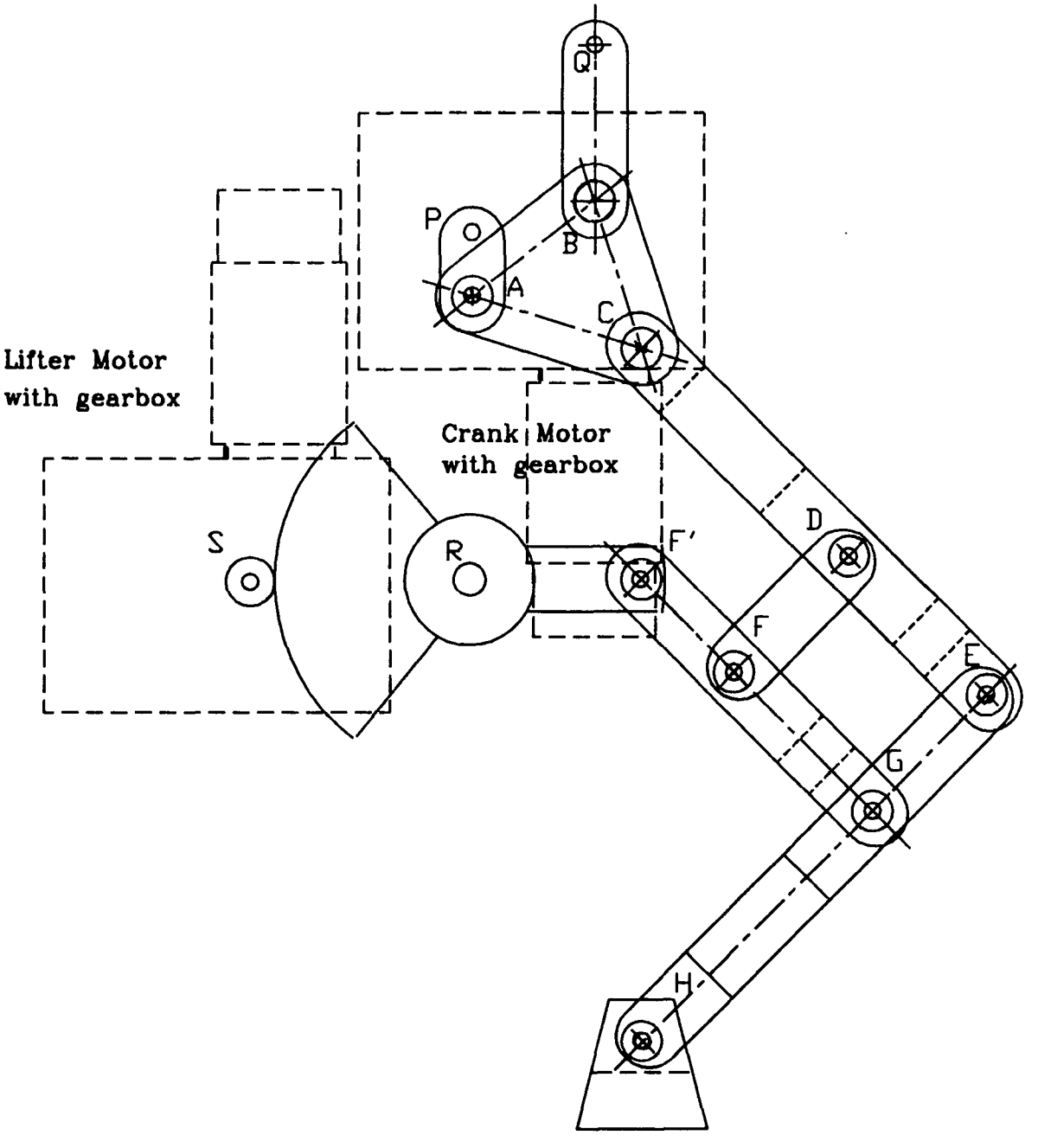

Q
P
B
A
C
Lifter Motor
with gearbox
Crank Motor
with gearbox
S
R
F'
D
F
E
G
H

Figure 1.4

compute the inverse kinematics equations for torque, force, displacement and acceleration at the intervals specified by the preprocessor. The resulting data is then tabulated and displayed graphically upon request.

The first three analysis performed on the leg in Figure 1. 4 were crank torques, joint forces, and joint displacements at points "C", "E", and "H". It was very important to determine the crank torque needed for normal walking and obstacle maneuvering. To find the maximum torque needed for normal walking, a force of 40 lbs was assumed at the foot (point "H"). This value was used to estimate the amount of weight (force) for the entire robot (240 lbs for a factor of safety of two), distributed over six legs. The DADS analysis for a time interval of 0.01 was found to give a maximum torque of 35 lb-in, as shown by the black line (in cgs system) in Figure 1. 5. This analysis prompted an idea to connect a torsion spring to point "Q" on the rocker link (see Figures 1.1 and 1.4). The torsion spring could be used to store the torque energy (created by the crank) while the foot was off the ground since less torque is required for this region of the stride cycle. Once the foot was again on the ground and its torque demand the greatest, the potential energy in the torsion spring could be released, resulting in a smoother torque vs. time curve (smoother walking motion). The torque vs. time curve for the addition of the torsion spring is shown by the blue line in Figure 1. 5. The addition of the torsion spring decreased the required crank torque to 21 lb-in; an appreciable change from the 35 lb-in needed when no spring was attached.

The crank torque was also analyzed during obstacle maneuvering. In this case, a force of 60 lbs was assumed at the foot since more force exists on the foot when climbing an incline surface. Figure 1.6 (in cgs system) shows the torque vs. time curve in black for a force of 40 lbs at the foot and no torsion spring on the rocker. The blue line of Figure 1.6 shows the torque vs. time curve for 60 lbs of force at the foot and the torque spring attached to the rocker. This analysis shows that approximately the same torque value can be achieved for 60 lbs with the spring as with 40 lbs without the spring. The proper spring constant "k" was determined through trial and error with DADS. The value of the spring constant was chosen so that the two peaks of the blue curve in Figure 1.5 would be even with one another. This meant that the torque cycle was as smooth (efficient) as possible. The value of "k" was found to be 1.02 lb-in per degree. This value was found using 40 lbs (normal walking force) instead of 60 lbs (climbing force) since the robot is under the

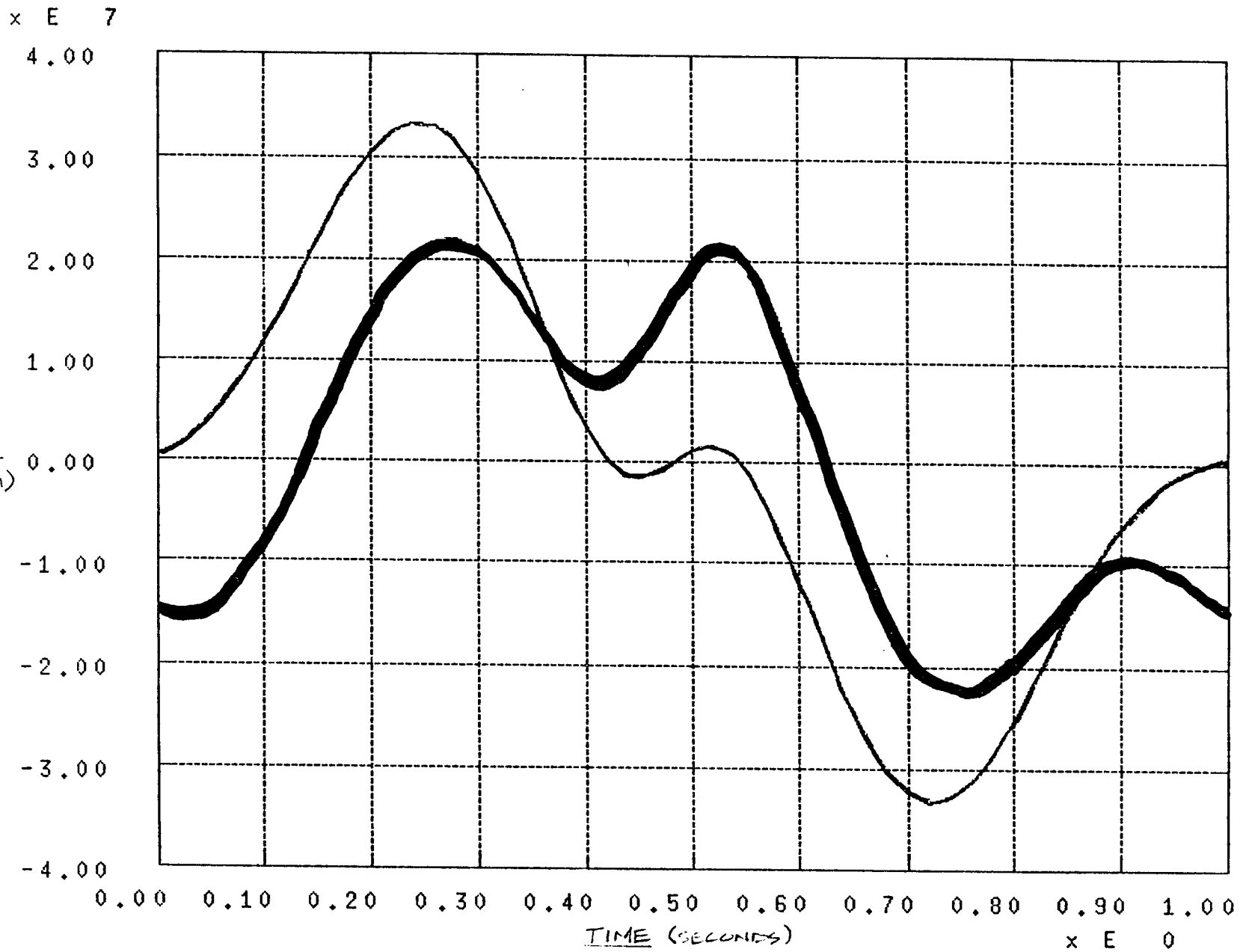


Figure 1.5

TORQUE VS. TIME CURVES

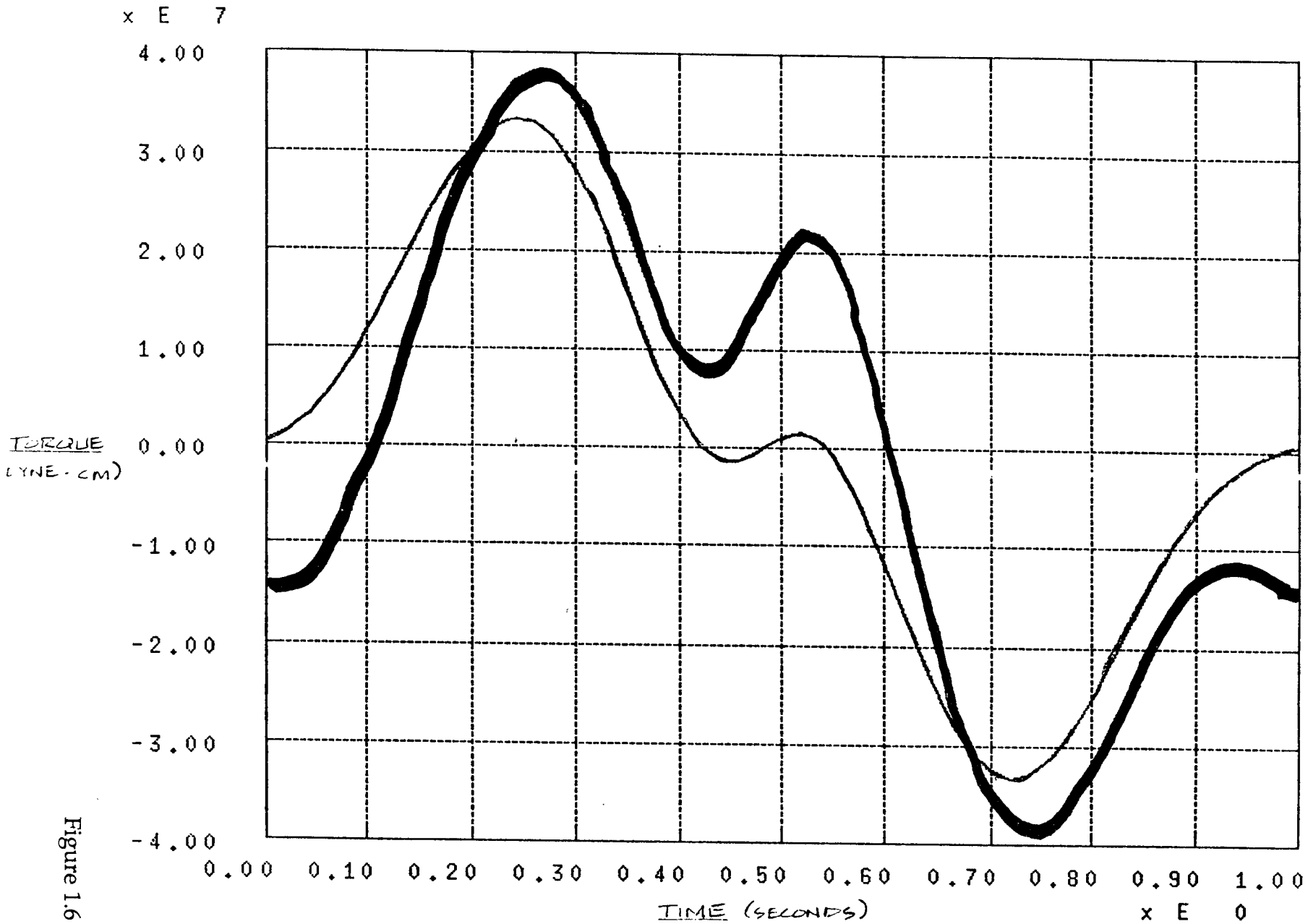


Figure 1.6

TORQUE VS. TIME CURVE
BLACK → 40 lbs ON FOOT W/O STRIN
BLUE → 60 lbs ON FOOT WITH

conditions of normal walking most of the time.

Figure 1.7 shows the DADS analysis for displacements at points "C", "E", and "H". Notice the inverted, amplified path of point "C" at point "H". The path drawn by point "E" shows the motion of the knee during normal walking. A force analysis was also performed on DADS to determine the forces at each joint of the leg. This information allowed proper bearings to be selected within acceptable safety factors. Most of the forces were approximately equal and the maximum force was found to be approximately 800 N. This occurred at point "F' " in figures 2 and 4.

Finally, through DADS and MATHPAC analysis, the equation for torque as a function of crank angle was determined. This equation was input into the robot's software, allowing a smoother walking motion.

Gearbox Design

Purchasing gearheads from the motor manufacturer was too expensive, therefore, gearboxes were designed and constructed. The leg crank and leg lift worm gears have different dimensions. The original idea was to design a different gearbox for each and minimize their dimensions. One eighth inch aluminum sheet was to be cut into appropriate sizes and then fastened together by L-brackets or welding. It was decided that this design was difficult and unnecessary. Two inch by four inch rectangular 6061-T6 aluminum tubing with one eighth inch wall thickness was selected for both gearboxes. This size provided the necessary strength and was considered the minimum thickness necessary to support the bearings. The leg lift gears required a box with a four inch length and the leg crank gears required a box with a three and one eighth inch length. Typical crank gearboxes and lifter gearboxes are shown in Figures 1.8 and 1.9, respectively.

Stainless steel shafting material was chosen for the gear shafts. The 303 grade was selected for its machinability, high strength, resistance to corrosion and relatively low cost. Brass couplers were chosen to attach the motor shafts to the worm shaft. Brass was selected due to availability, low cost, and machinability. Past problems involving set screws in gears, and couplers slipping on shafts, prompted the drilling and pinning of all the gears and couplers to the shafts. The gears were first aligned and locked in place with a set screw. The hubs of each gear were then drilled slightly undersized for the diameter of the hardened spring steel pins.

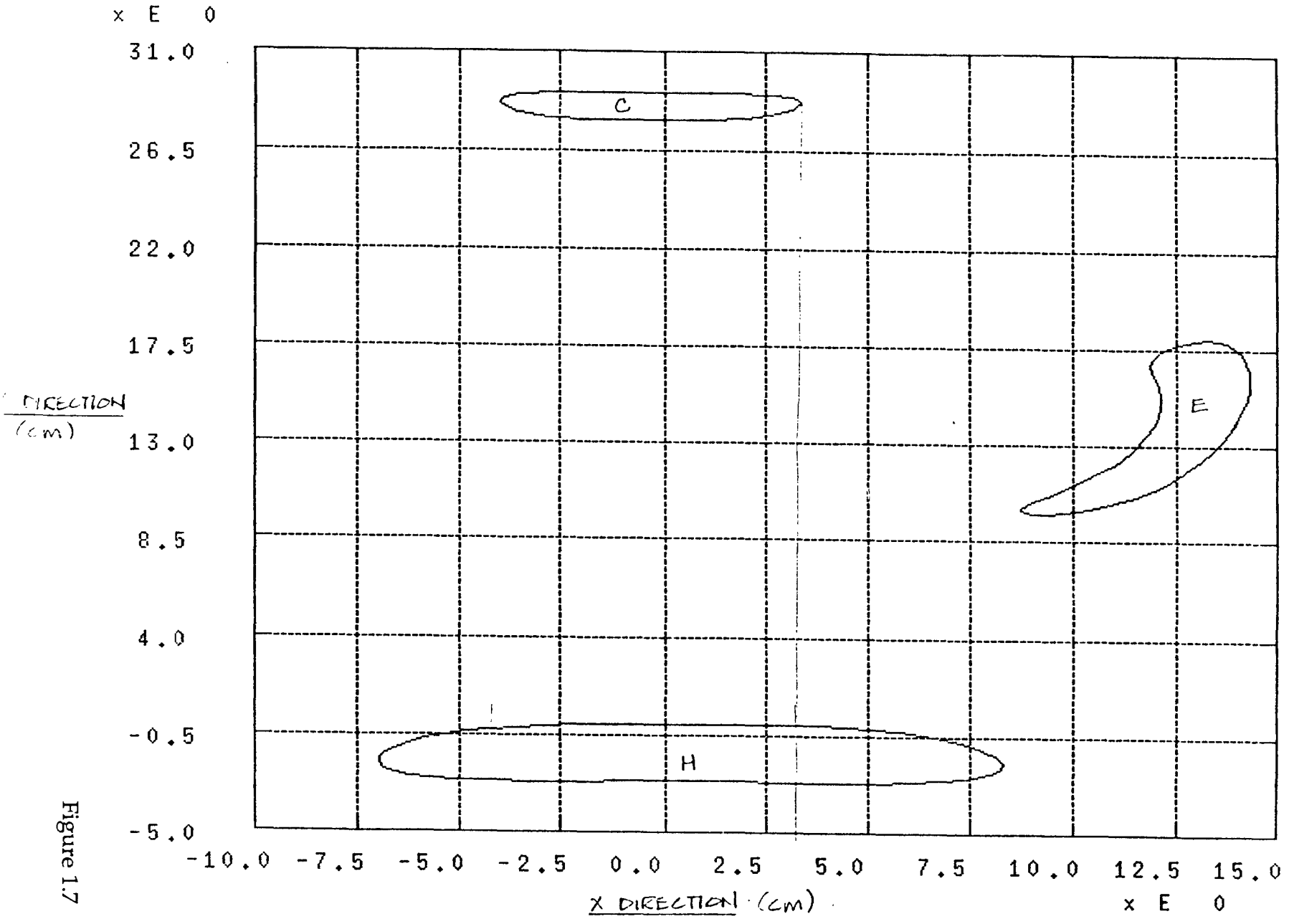


Figure 1.7

X VS. Y DISPLACEMENT

C → COUPLER PATH
E → KNEE PATH
} SEE FIGURE 1.4

FIGURE #8; LEG CRANK GEARBOX

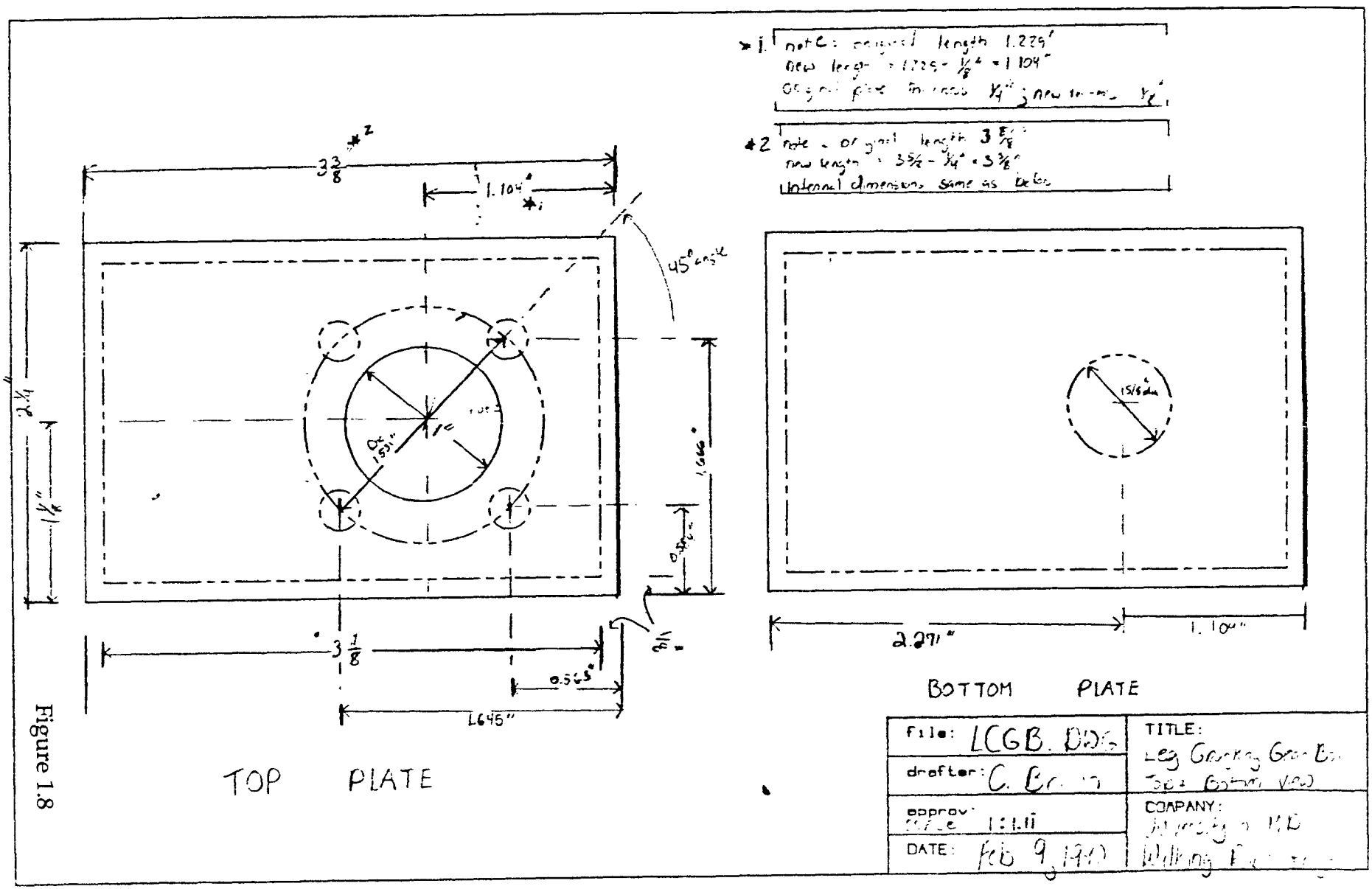


Figure 1.8

FIGURE #9 : LEG LIFTER GEARBOX

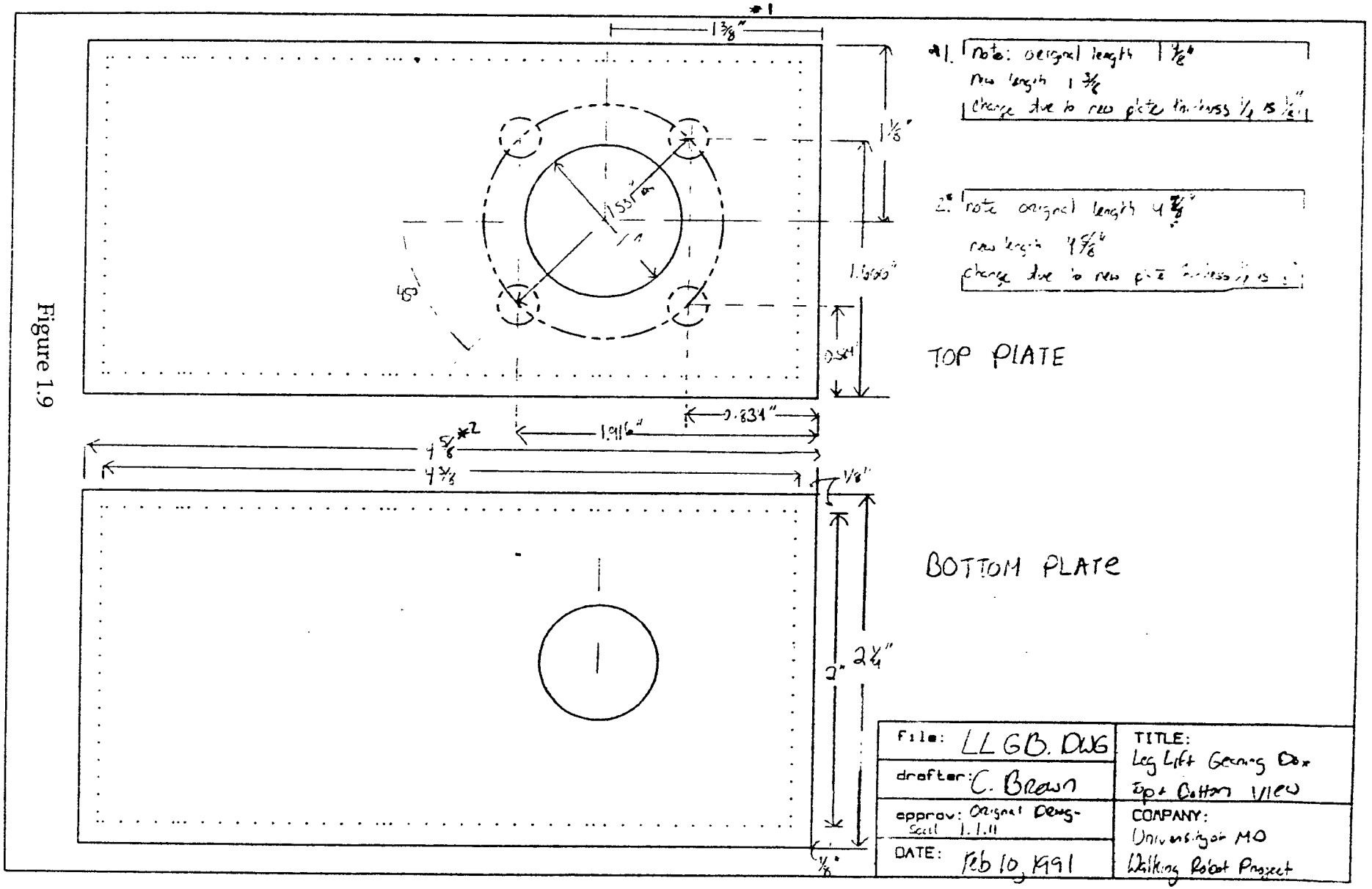


Figure 1.9

Gear Selection

Due to space constraints, the motors could not be mounted between the support plates. To minimize the width of the overall assembly, the motors were mounted flush against one of the plates. The following items had to be considered when selecting the type of gears best suited for this application:

-The torque from the motors needed to be transmitted between non-parallel, non-intersecting shafts.
-The high torque and shaft speeds from the motors had to be reduced to drive the crank and further reduced to drive the leg lift mechanism.
-Backdriving of gears was a problem in previous robot designs and must be considered.
-The gears had to be easily accessible and adjustable in case there was a problem.

For these reasons a worm and worm gear combination was selected. There is often a large difference between the pitch diameters of the worm and worm gear. This difference was considered during the selection of the gear material. The worm must rotate many times to rotate the worm gear once, therefore, in order for wear to occur equally between the worm and worm gear, the worm was made of hardened steel and the worm gear was made of brass. The gear ratio for the crank gearbox was selected to be 35:1, while the lifter gearbox ratio was 60:1. All gears were purchased from Boston Gear, Inc.

Motor Selection

Thirteen motors were were needed for this design, therefore, they were the most expensive components of the entire robot design. For this reason, special attention was given to their selection. Based on durability, low weight, small size, high torque, and cost, 14202 series Pittman DC servo motors were selected. Motor specifications are listed in the appendix.

Two motors were required for each leg. One motor drives the crank, the second motor drives the leg lift mechanism. Originally, MicroMo Corp. motors

were selected with gearheds attached. These motor/gearhead combinations, however, were nearly three times the price of the Pittman motors and it was discovered after closer examination that the gearheads would not withstand the large torques of this design.

Another consideration in motor selection was the the choice of encoders. Hewlett-Packard encoders were supplied and attached by Pittman in order to reduce complexity of integration and assembly. Due to cost constraints, the encoders selected were incremental not absolute. To provide absolute positional data to the motor controllers, the encoders were augmented with contact microswitches located at each extremes of the pie shaped lifter gears; on the bottom of each foot; and at an extreme position of the rockers of each leg assembly.

Leg Links

The leg links are the load bearing members of the crank and rocker and of the pantograph mechanisms. The crank and rocker consists of a steel coupler plate, a steel crank bar, and an aluminum rocker bar. The pantograph mechanism consists of four aluminum bars. These materials were specified and machined so as to provide adequate strength under impact conditions while maintaining very close hole-to-hole and part-to-part tolerances. To facilitate this, a highly rigid and precisely controlled machine tool was required. A computerized, numerically controlled (CNC) vertical milling station was provided to the students by a sponsor.

Design Optimization

The design of the five aluminum links was optimized for simplicity of construction. This was achieved primarily by incorporating common design details in each of the links. The benefits of using typical details manifest themselves through all stages of the construction process. Materials were economically obtained due to the quantity pricing advantage of homogenous specifications. Drawings were easily created through use of CAD software. Programming the machine control was done efficiently. A common fixture was used to clamp the links during machining. The automated machining itself was performed with only six tools. The mechanism assembly procedure was also simplified by the modular part structures.

Materials

The material for the links was originally purchased in the form of two 12 foot bars of 1" square solid 6061-T6 aluminum.

Drawings

The detailed part drawings were made with CAD software. The modular parts allowed use of blocks for part details, which were copied and modified for each link drawing. Further use of CAD blocks provided an analysis technique for determining internal clearance conditions. Datum dimensioning was employed using these criterions. A rough block drawing of the fixture profile was used to determine the logical position for datum placement. This method afforded a clear understanding of the cutter path with respect to the fixture and vice. The dimensioning technique facilitated data extraction for programming the machine control. CAD drawings of the links are included in the appendix of this document.

Programming

The machine control was programmed remotely through use of a macro library. The macro language translates written commands into machine language according to a set of definitions. The modular part design allowed extensive use of subroutines for program details. The goal in programming the machine controller was to describe the cutter path so as to affect the specified cuts while avoiding interference (collision) with the fixture and vice.

Programming the control also required understanding of the actual cutting process. Proper spindle speeds and feed rates were calculated and specified for an array of drills and reams and for an end mill under various loading conditions. This process is highly empirical and requires fine tuning during the machining process. A sample set of programs (in non-compiled form) are included in the appendix of this document.

Fixture Design

The function of the fixture was to hold the aluminum bars within the vice while machining. The design of the fixture had three major constraints. The fixture was required to index off of the machine table in order to define the part position with respect to the control's coordinate system. The fixture had to be sufficiently rigid so as not to deflect under the entire clamping force of the vice. The size of the clamp blocks themselves was also critical. Their width (parallel to part length) was restricted to being small enough so as to allow maximum cutter path flexibility around the part. The depth of the blocks had to provide tool clearance between the vice and the part. This is indeed the very purpose of the fixture: to provide the tool with free access to the small part in a large vise.

The fixture consisted of five components, four of them steel. The base, a thin plate, was welded to an indexing flange and a clamp block. The flange was flushed with the vice corner. The second clamp was kept free to allow for material compression and variation. The fifth component was an acrylic riser used to protect drills as they clear the bottom surface of the piece. This part was replaced when, due to wear, it could no longer provide a flat surface on which to rest.

Tooling Selection

Milling, drilling, and reaming were the machining operations specified by the parts' designs. All holes required tight diametral tolerances, hence a tool staging procedure was employed. The first operation is a center drill. This prepares the part for the drill through operation in that there is less tendency for the drill bit to "walk" as it enters the piece. The drill diameter was 1/64" undersize to ease the reaming stage. A ream cuts holes far more accurately than a drill.

The interior slotting and exterior contouring work was performed with an end mill. The parts were designed to be milled with only one tool. The tool selected was designed for use on aluminum, the chips of which has a tendency to lodge in the flutes of tools intended for steel.

Machining Process

The process of machining the aluminum links began by cutting the aluminum bar into rough part lengths with a cut-off saw. It was determined that a blade with widely spaced teeth, running at high RPMs, provided the best quality cut for the aluminum grade utilized. Six lengths were cut for each link design, one per leg, except for short link DF. Three lengths were cut to the rough length of the short link, which were then cut lengthwise by a bandsaw to form the required six pieces. All 30 aluminum pieces were then deburred on a light grit grinding wheel.

Next, the links were prepared for their machining cycles. The most important factor in this preparation was the requirement that the majority of the links be machined on two perpendicular faces. Given the available equipment, this factor called for the pieces to be turned 90 degrees at some point during the machining cycle. This is not a trivial step, as the machine is computer controlled and the piece must therefore be replaced in the same location from which it was removed. To accomplish this, reference lines were drawn on each piece. These marks were drawn in locations free of external details and alligned with a scribe line on the center of the welded fixture clamp. This placed the pieces into a known position with respect to the machine coordinate system. Tolerances between the two faces were designed to be such that when the pieces were turned 90 degrees, positioning by eye was sufficient.

The applicable program was then begun. When it was fully debugged, the process from this point on was trivial. Although a great deal of time was spent debugging the programs, the final versions were all but autonomous, requiring only observation and adjustment of the coolant nozzle. This process lends well to high production quantities. Through modular design and use of subroutines in programming, debugging was simplified and thus production efficiency improved through reduced setup time.

Section II: Body

The purpose of the body group was to meet certain objectives in the process of completing this component of the walking robot:

- To design a platform capable of retaining all major components of the robot
- To minimize the deflection, weight, cost and complexity of construction.
- To maximize the toughness and ease of manufacture.
- To create a frame capable of causing the robot's turning motion
- To build an assembly used for competition in the hockey puck event

Frame

The overall design dictated the outline and geometry of the chassis. The design is as follows:

Two triangular (isosceles) frames containing a leg mounting boss at each apex are superimposed in a "star of David" configuration. The two frames rotate about a common axis passing roughly through the center of mass of each frame. Several types of chassis construction were considered. The original design, a trussed space frame, was deemed too complex due to the large number of mitered joints and the interplay of component members between the two triangles. It was therefore decided to utilize a platform chassis (Figure 2.1) operating in a single plane for each frame, thereby eliminating any possible interference between the triangles.

A platform chassis may be constructed in several ways and out of several materials. In any design project there are two possible approaches. First, there is a design which is easy to conceptualize but difficult to manufacture. Second is a design that is easy to manufacture but difficult to conceptualize. A ladder frame of two side rails and several straight cross members of generous wall thickness was considered. This is easy in concept but labor-intensive to manufacture. This is due to the fact that the shape requires a large number of cut and mitered joints at odd angles. The other approach is a unitized combination of two parts, a flat skin and a reinforcement panel with stamped-in stiffening ribs. Once spot welded together, they produce a single platform that is both thin and extremely rigid. A production rate of several thousand units per hour is possible for this design approach. The excessively high

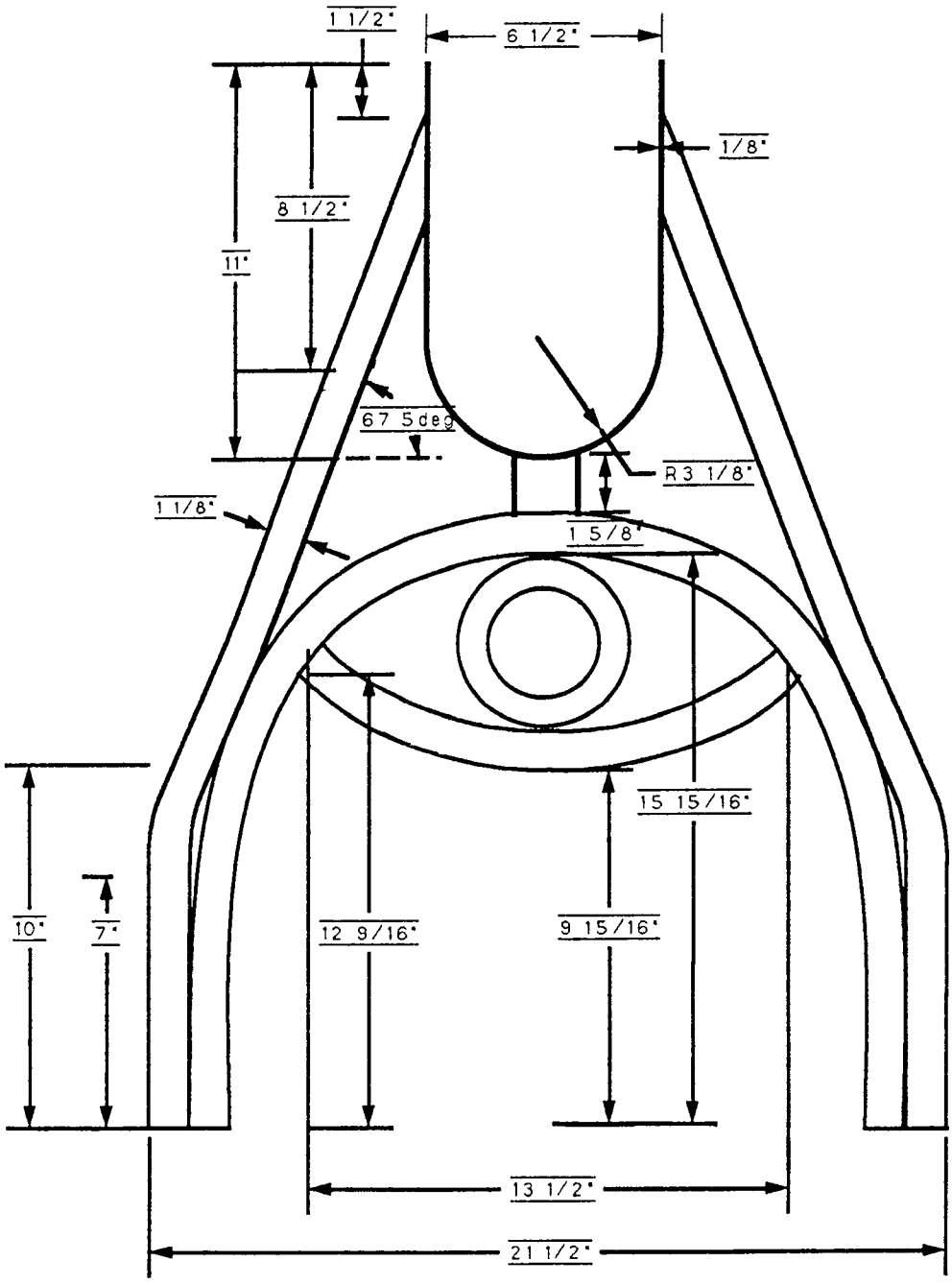


Figure 2.1: Frame Dimensions (one frame)

production rate along with moderately high tooling costs were beyond the scope of this project. Therefore, A fiberglass version of this concept with hand laid foam core stiffening ribs was briefly considered. This consideration was rejected due to time consuming, tedious load analysis required for placement of reinforcements. This process would work if sufficient time and expertise were available during the design stage.

The final design was a combination of the ladder frame and the unitized frame. This modified frame contained two thirds less material than the original ladder frame design allowing the use of steel tubing instead of aluminum. A modified ladder frame of an irregular shape can be made easily if the members can be formed into the desired shapes by simple bends. This would reduce the number of joints. The material choice for this application was round thin wall tubing. The strength lost due to the reduction in wall thickness (as compared to traditional ladder frames) is compensated by the reduction in the number of fastened joints. An additional strength is created by the installation of thin steel sheet reinforcements. These steel sheets are installed at the top and bottom of the main frame by spot welding the sheets to the thin wall tubing. The spot welding can be accomplished where ever opposing access is available on the outside diameter of the tube for electrode placement. Spot welding has the added advantage of easy automation and preservation of the rust resistant capabilities of galvanizing. This design provides low tooling and material costs, ease of automation (bending and welding) and a large or small volume production. But, since the spot welder was not at the disposal for this project the steel sheets were installed by using pop rivets.

The overall design of the robot was extremely sensitive to any backlash. Specifically, the bearing hub would not have functioned adequately in a presence of any backlash. This sensitivity was demonstrated during the initial assembly of the two chassis. It was noticed that with one tapered bearing seat removed, a 0.001" assembly clearance was magnified to 1/8" at the points of leg mounting. The tapered bearing seat was reinstalled, and a static load test was performed. The equipment used in this test was a vernier calipers, a surface plate, and a ten pound weight. Three trials were run with an average deflection of .024". The maximum and the minimum deflections were .030" and .018". The chassis's performance proved to be robust.

There were two incidents during the competition that proved the strength and durability of body's robust design. The first and most serious was the final

assembly of the robot with only three of the five bolts installed to hold the bearing hub to the upper chassis. The second was a large clamp which was applied to prevent relative motion of the frames during handling. This is normal except in one case in which the 3/4 inch spacer was not installed between chassis. The result was that the chassis were forced together on one side to the point of contact. Fortunately no permanent damage was done to the chassis or bearing hub. The repair consisted of simply prying the chassis apart to the required 3/4". Since it has proven itself so well, the body will remain unchanged as the robot is brought to competition performance.

Turning Mechanism

The design of the turning mechanism for the body incorporates several design criteria. The mechanism is simple and was easy to build, easy to assemble and disassemble, light weight, strong enough to withstand the loads applied to it and powerful enough to turn the body under dynamic loading conditions. The final design takes into account each of these constraints.

In order to keep the design simple it was decided that it should consist of only four parts: the hub, bearings, shaft and motor (see Figure 2.2). The hub, which houses the bearings, was machined out of one piece of aluminum to insure light weight and high strength. Two tapered roller bearings were press fit into the hub. The shaft is locked into the bearings with a 7/8" nut. The motor shaft was to be inserted directly into the turning shaft and held in place by setscrews to increase efficiency and simplicity. However, we were forced to change this element of the design for reasons which are outlined further on.

Weight reduction was largely achieved through the use of aluminum in the hub. The shaft is steel due to strength requirements imposed by overcoming the inertia of the body, and makes up the bulk of the weight in the turning mechanism. Use of composites, alloys or other materials for further weight reduction is recommended for any future design.

Strength requirements dictated the path of several design features. Five through bolts anchor the hub to a 3/16" thick steel plate that was welded to the frame of the body. Bearing strength was achieved through the use of automotive tapered roller bearings. These bearings provide more than adequate strength as well

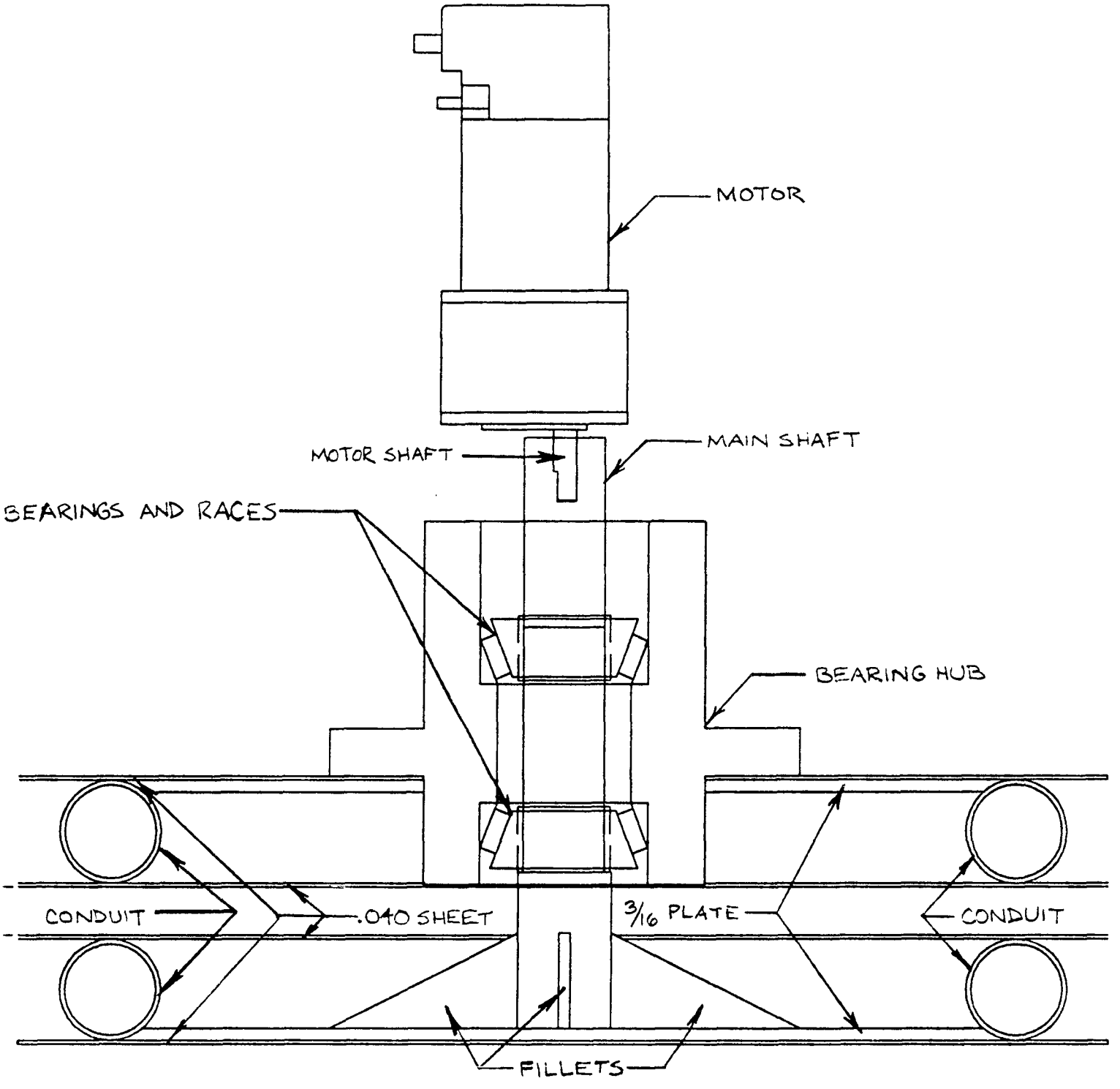


Figure 2.2: Original Turning Mechanism Design

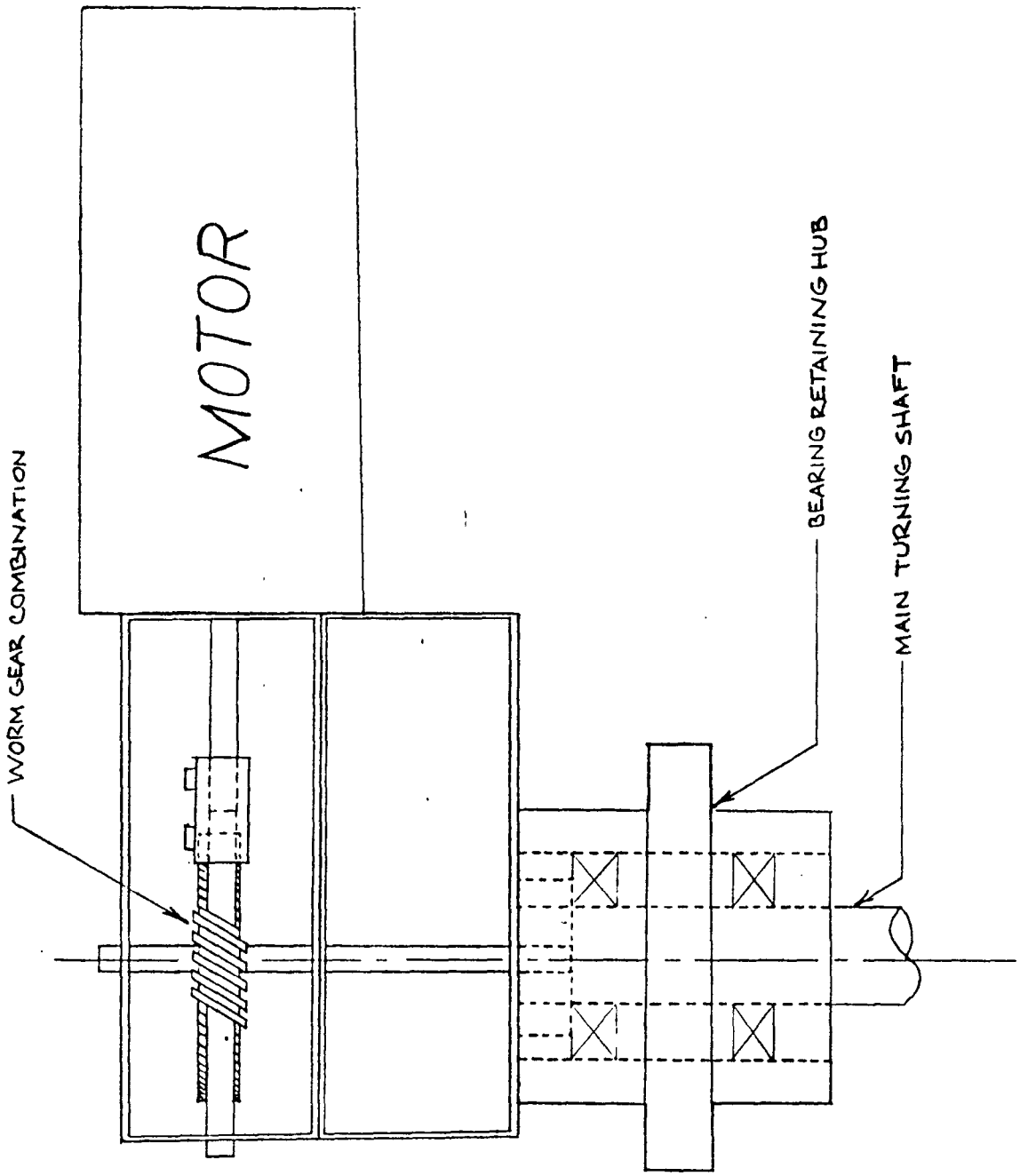


Figure 2.3: Temporary Turning Mechanism Design

as zero backlash. The damping effect of the tapered bearings is considered to be useful in overcoming the angular momentum of the body. As mentioned above, the shaft is made of steel which is of sufficient strength to handle the applied loads and torques. The shaft was welded onto a 3/16" inch steel plate that was in turn welded into the frame of the body. Triangular 1/8" thick steel fillets were welded to the plate and shaft to provide extra bending moment strength.

The motor shaft was originally designed to fit directly into the turning shaft. The motor, a Pitman model GM9414 with 24V nominal voltage, a 127.7:1 gear ratio, and a 100 CPR encoder, was capable of producing the power necessary to turn the body against the moment of inertia generated by the weight of the frame and legs. However, the motor's reduction gears were not able to withstand the applied loads and were destroyed. Under these last-minute conditions, we were forced to use one of the extra leg lifter motors and gearboxes (Figure 2.3) as a substitute turning motor assembly. Later, we will replace this temporary system with the original turning assembly, using a new gearbox with stronger components. When the microprocessors function correctly, they will insure that the turning motor does not turn the body too far, causing the legs to hit each other and the motor to stall. This way, the motor will not spend much time working at stall torque, and lesser demands will be made of the gears.

Hockey Stick Assembly

The performance requirements for the hockey puck event of the competition necessitated a design with certain criteria. The hockey stick assembly had to be able to drag or push a puck continuously through the event. The event had two major influences on the design of the hockey stick. First, the assembly had to be able to turn independently of the body. Second, the assembly had to have freedom to move in the vertical direction.

The purpose of the first design criteria was to assure that as the body turns and moves forward or backward, the hockey stick can keep the puck in the desired position. The second design criteria is incorporated in the design of the hockey stick to allow the blade to have continuous contact with the ground and puck as the body rises up and down in its normal movement path.

The hockey stick assembly is composed of two links and a blade. Only the

blade has contact with the puck. Link 1 is vertical and connects to the blade, giving the stick the proper height to connect to the motor. Link 2 is horizontal and connects to link 1 and to an aluminum joint. Link 2 acts as a displacer for the blade, giving it an arcing motion, which causes the puck to return to center. The aluminum joint, and therefore the links and blade, is turned by a stepper motor. The motor shaft is fitted into the aluminum joint and is reinforced by a set screw. The motor is mounted to the rear leg housing by an L-bracket.

The first design criterion, control of the puck's direction of movement, was satisfied by the turning motor, which is controlled by an operator through a tether. The hockey stick was installed on the rear leg housing for maximum operator visibility. The second design criteria was satisfied by the use of the aluminum joint. With the rise and fall of the body during its normal walking pattern, the hockey stick assembly will also rise and fall. The joint allows the hockey blade to maintain contact with the ground by giving the links and blade freedom to rotate about an axis perpendicular to the links and parallel to the ground. The weight of the assembly causes it to rotate about the joint, thereby keeping constant contact with the ground and puck, providing continuous control by the operator of the hockey stick assembly.

In sum, the body group created a frame capable of meeting the objectives stated above. A few changes may be made to the basic design between now and next year's competition, but overall the robot body proved its worth in its satisfactory performance on the field of competition.

Section III: Electronic Hardware

The Hardware Group was responsible for the design of the computer 'brain' for the walking robot. This computer controls a total of 13 motors, each with an optical encoder to monitor it's position, and monitor at least 26 switches. Since the robot must be completely self-contained, the electrical power for the robot comes from on-board 12 volt batteries.

The decision of the type of computer system to use took into account the cost, computational power, and the ease of external device (motor) control. A PC is widely familiar computer, but lacks easy device control. An AT is essentially a PC, but with more computational power. A microcontroller is a "computer on a chip"– it is meant for applications that require easy device control. But a complex task that a PC or an AT can handle in stride, the microcontroller comes up deficient. There are five possibilities for controlling the motors:

1. PC to Motors
2. AT to Motors
3. Microcontroller to Motors
4. PC to Microcontrollers to Motors
5. AT to Microcontrollers to Motors

The first was eliminated because although the hardware connection would be simple, the software would become extremely difficult. The computing power required by the PC would be more than it could supply. The second choice was eliminated for the same reasons. Microcontrollers directly to the motors was eliminated because of the difficulties encountered when it was attempted previously. The increased computational power provided by the AT does not overcome the increase in cost. The microcontrollers are designed for motor control applications. They can easily be used to control two motors and keep track of the encoder data, making it useful to control a single leg. A PC commanding microcontrollers controlling the motors was the system that best fit the requirements. Figure i.1 is the basic hardware configuration.

The next task was deciding how to communicate between the PC and the microcontrollers, and between the microcontrollers and the motors. The first decision was the choice of serial or parallel communication between the PC and the microcontrollers. The software needed for serial communications is greater than for parallel, while the hardware is about the same. It was decided to use parallel

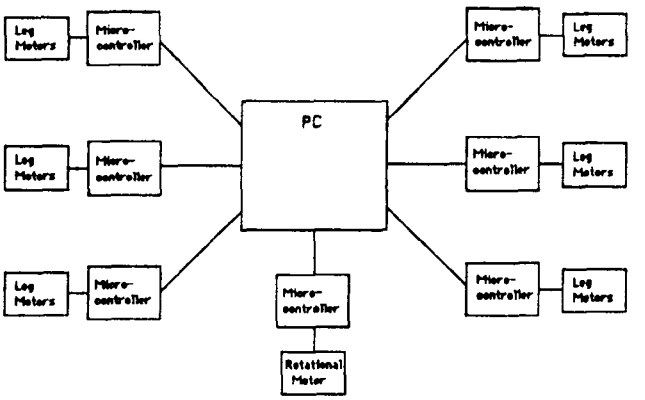


Figure 3.i

communications. With parallel communications, comes the problems of bus protocol. A shared dual port memory allows a separate common memory between the PC and each microcontroller that both the PC and the microcontroller can access at the same time. Additional hardware and software is kept to a minimum. The dual port memory can be directly connected to the microcontrollers address and data buses. Address decoding is required on the PC side so that it can access the correct bank of memory.

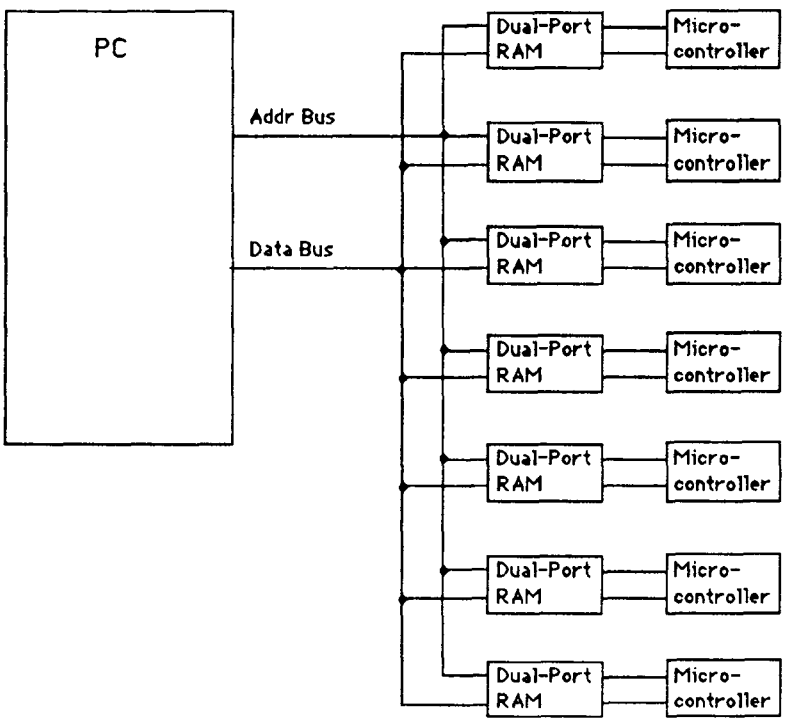


Figure 3.ii

The microcontroller to motors connection is outlined in the application notes for the 80C196KB. The microcontroller sends the direction and speed to the motor driver circuit. The motor driver circuit is needed to amplify the signal since the motors use more power than the microcontroller can supply. The circuit uses the pulse width modulated signals from the microcontroller to control the speed of the motor.

The design philosophy was to keep the circuitry as simple as possible, within the specifications outlined above. Part of the simplicity was the modularization of the components. Each individual sub-system was built and tested independently of all the other sus-systems to localize problems before the robot was completely assembled.

Part I
The PC interface

The main processing unit of the hardware design is an IBM PC motherboard. The PC controls the legs by communicating with the 80C196 microcontrollers. This communication is accomplished through a Dual Port Static RAM (SRAM) that is shared between the PC and the microcontroller. The PC also handles interfacing with the voice control and the sensors.

The PC shares one dual port SRAM with each of the seven microcontrollers. The PC uses the right side of the dual port and the microcontroller uses the left side. These dual port SRAMs are mapped into the PC memory space. The dual ports are mapped into the memory space assuming the PC has it's maximum RAM memory (640Kb). The address space between the 640Kb RAM maximum and the 1Mb top of memory was explored for a suitable area to map the dual ports. A 248Kb block was found starting at address C0000h. Each dual port is 2Kb, therefore at least 14Kb is needed. A large enough section of the 248Kb block for all the dual ports to be mapped contiguously was found at address CC000h through CFFFFh. Figure 3.1.1 shows the complete memory space of the PC (Duncan, 87). This part of the block is used to simplify address decoding. Figure 3.1.2 shows the address decoding circuit. The circuit uses the nine high order address lines and AEN (Address Enable) to generate a chip select signal when the appropriate memory block is accessed. The signals are taken from the PC's expansion bus (labeled PC Bus in the figure). A table of addresses associated with each signal is also included in the figure. Once the dual port mapping was established, the connections between the PC and the dual ports needed to be defined. The address and data lines connect directly from the PC to the dual port. The read and write signals from the PC do not correspond directly to read and write signals on the dual port. The dual port has one signal indicating read or write (R/$\overline{W}$) and one signal for enabling the output ($\overline{OE}$). The dual port function associated with these signals while the chip is enabled are:

R/$\overline{W}$	$\overline{OE}$	Function
L	X	Write to dual port
H	L	Read from dual port
H	H	High impedance state

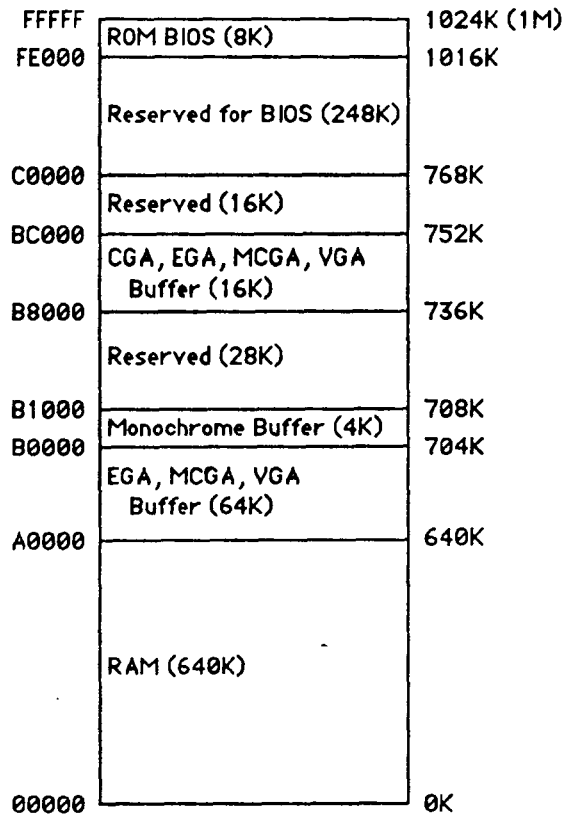
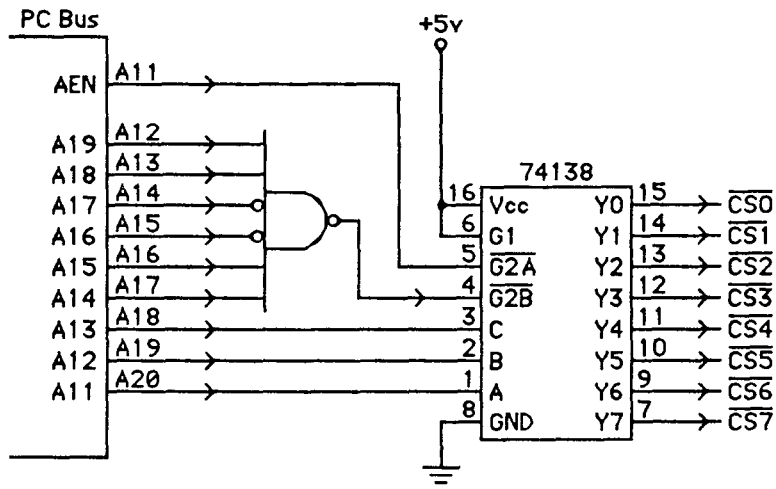


Figure 3.1.1



Chip Select Active	Address Block (Hex)
0	CC000-CC7FF
1	CC800-CCFFF
2	CD000-CD7FF
3	CD800-CDFFF
4	CE000-CE7FF
5	CE800-CEFFF
6	CF000-CF7FF
7	CF800-CFFFF

Figure 3.1.2

The PC functions associated with the memory read signal (MEMR) and the memory write signal (MEMW) are:

MEMR	MEMW	Function
H	L	Write to memory
L	H	Read from memory
H	H	No read or write

The dual port functions and the PC functions coincide if the R/W signal is connected to the MEMW signal and the OE signal is connected to the MEMR signal. The timing diagrams and calculations show that these signals meet the required timing conditions. The diagrams and calculations appear at the end of this section.

In addition to the read and write signals, the dual ports generate two signals that the PC must monitor. They are busy (BUSY) and interrupt (INT). If the busy signal goes active, the dual port is requesting a wait state. A wait state is when a device can not respond to an access by the processor and the processor allows extra time for the device to respond. This signal is connected to the I/O Channel Ready input on the PC. This input signal is used to request wait states. If the interrupt goes active then there is information in the dual port that the PC needs to read. This is connected to one of the interrupt request (IRQx) lines on the PC. These connections are shown in Figure 3.1.3. The busy and interrupt signals are not quite as simple to connect as described. Since there are seven dual ports, there are seven busy signals and seven interrupt signals that need to be monitored. The circuit that accomplishes the combination of the seven busy and seven interrupt signals into two signals is shown in Figure 3.1.4. The circuit sends a busy signal to the PC if any of the dual ports generate a busy signal. The PC does not need to know which dual port needs the wait state. If any of the dual ports generate an interrupt signal, an interrupt signal is sent to the PC. Once an interrupt is received, the PC needs to know which dual port sent it so that the information can be read. This is accomplished by connecting the interrupt lines to an input port of a Programmable Peripheral Interface (PPI) that is connected to the PC. When the PC receives an interrupt, the PPI port contains the status of the interrupt line. By reading this port, the dual port that generated the interrupt can be determined. Since the complete configuration of the PC motherboard is not known, a jumper block is used to select

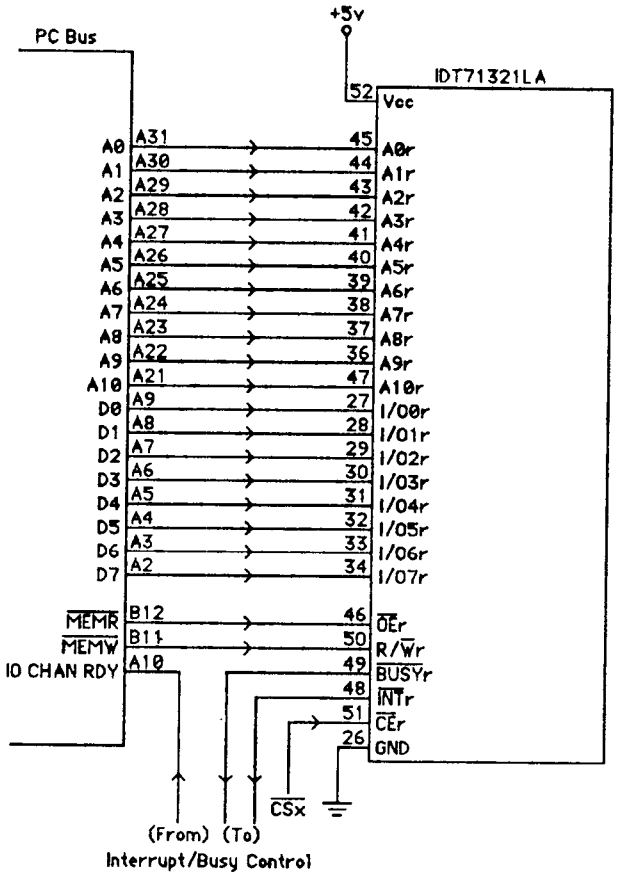


Figure 3.1.3

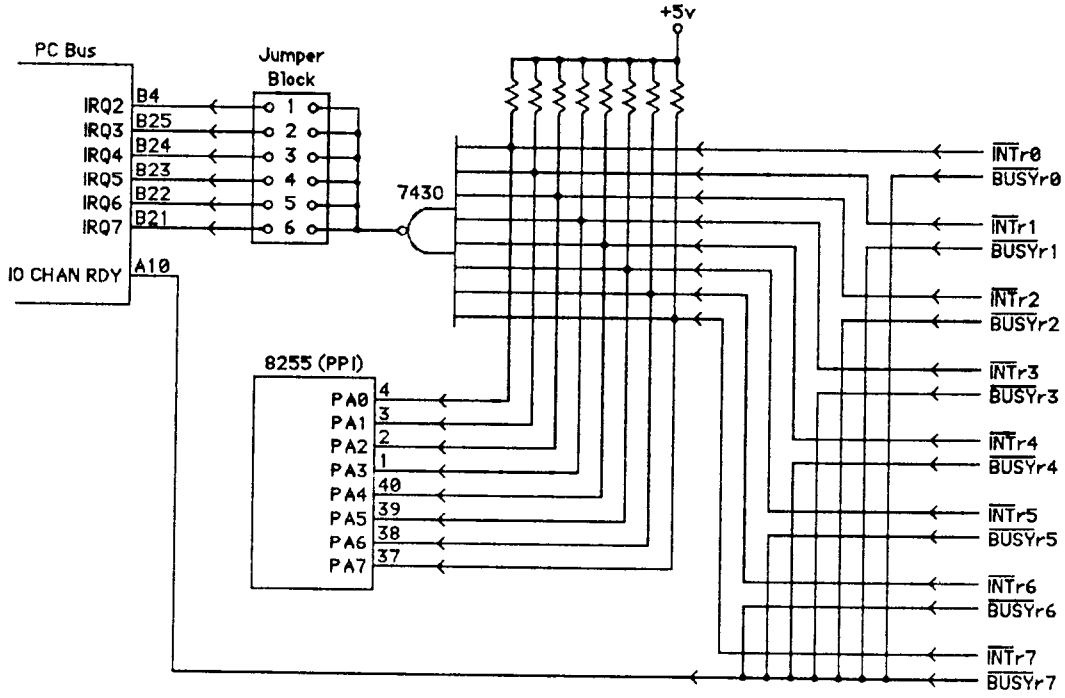


Figure 3.1.4

which interrupt request line is used to inform the PC of an interrupt. The timing diagrams and calculations show that the busy and interrupt signals meet the required timing conditions. The diagrams and calculations appear at the end of this section.

The PPI is connected to the PC and is used to interface with several devices. The first use has already been discussed, the monitoring of the interrupt lines from the dual ports. Other uses include interfacing with a voice recognition circuit and sensors. The voice recognition circuit has up to eight outputs to indicate commands. One port of the PPI can monitor these eight lines for commands at a time when voice control is desired. The PPI can also monitor sensors. Optical sensors generally have one output signal, therefore up to eight sensors can be monitored with one port. The PC to PPI connections are shown in Figure 3.1.5.

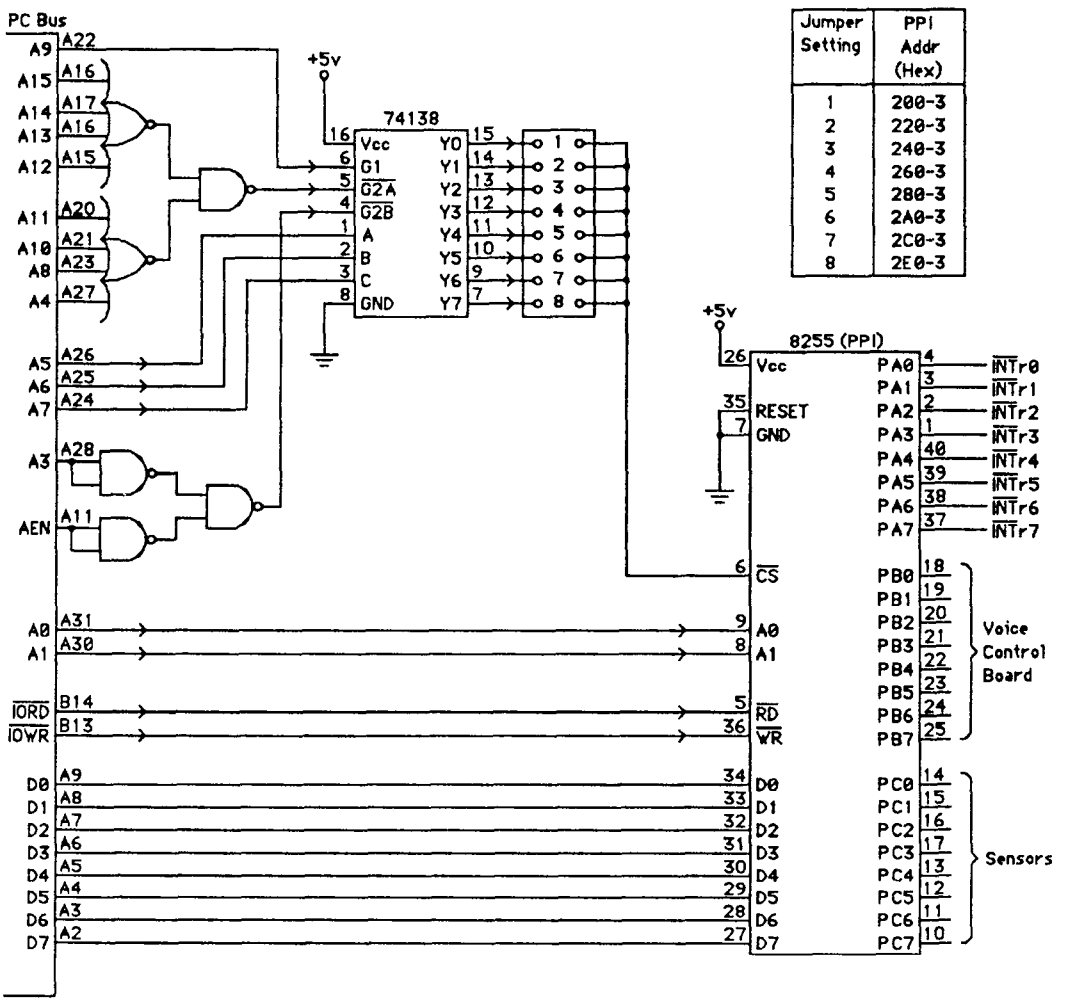
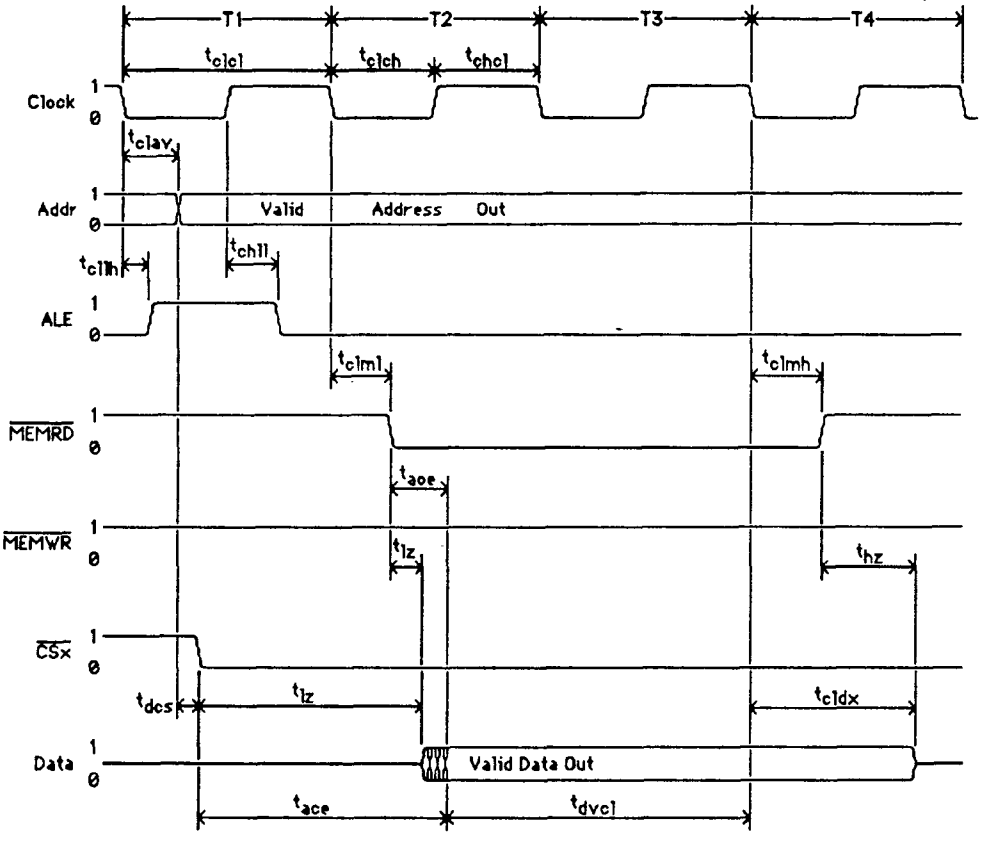


Figure 3.1.5

Timing Diagrams and Calculations:

This section contains the timing diagrams and calculation as mentioned previously in the text.

Read Cycle:



Read Cycle Calculations:

Two conditions must be satisfied for the read cycle to function correctly. All numerical values are in nanoseconds. A list of parameters is included at the end of this section.

1.) Valid data must be output by Dual Port before data is read by PC:

valid data = $t_{clcl} + t_{clml} + t_{aoe}$

data required = $3t_{clcl}$

data set-up time = t_{dvcl}

(data required) - (valid data) > (data set-up time)

$$(3t_{clcl}) - (t_{clcl} + t_{clml} + t_{aoe}) > t_{dvcl}$$
$$2t_{clcl} - t_{clml} - t_{aoe} > t_{dvcl}$$

Minimize the left side and maximize the right side:

$$2(200) - (35) - (40) > (30)$$
$$325 > 30$$

This condition is satisfied.

2.) Valid data must be output by Dual Port before data is read by PC:

valid data = $t_{clav} + t_{dcs} + t_{ace}$

data required = $3t_{clcl}$

data set-up time = t_{dvcl}

(data required) - (valid data) > (data set-up time)

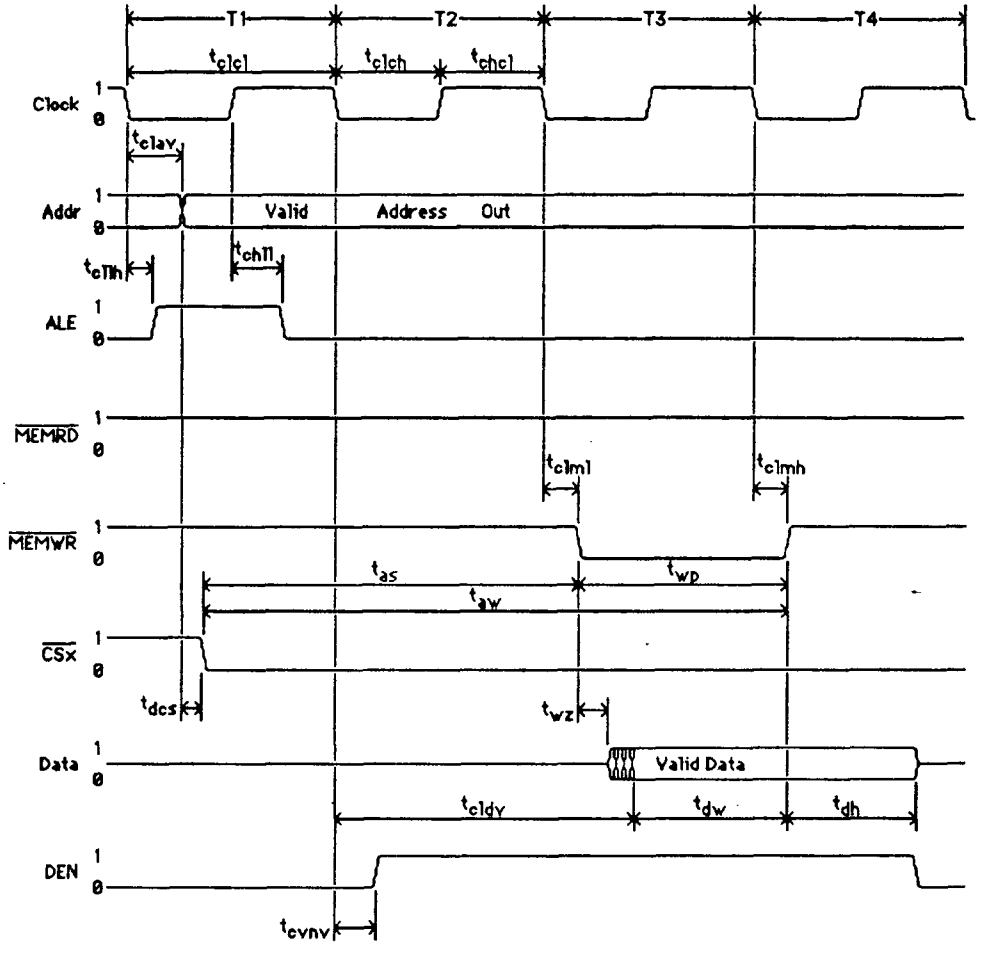
$$(3t_{clcl}) - (t_{clav} + t_{dcs} + t_{ace}) > t_{dvcl}$$
$$3t_{clcl} - t_{clav} - t_{dcs} - t_{ace} > t_{dvcl}$$

Minimize the left side and maximize the right side:

$$3(200) - (110) - (35) - (70) > (30)$$
$$385 > 30$$

This condition is satisfied.

Write Cycle:



Write Cycle Calculations:

Three conditions must be satisfied for the write cycle to function correctly. All numerical values are in nanoseconds.

1.) Valid data must be output by PC before data is read by Dual Port:

valid data = $t_{clcl} + t_{cldv}$

data required = $3t_{clcl} + t_{clmh}$

data set-up time = t_{dw}

(data required) - (valid data) > (data set-up time)

$(3t_{clcl} + t_{clmh}) - (t_{clcl} + t_{cldv}) > t_{dw}$

$2t_{clcl} + t_{clmh} - t_{cldv} > t_{dw}$

Minimize the left side and maximize the right side:

$2(200) + (10) - (110) > (30)$

$300 > 30$

This condition is satisfied.

2.) The time from chip enable to data required must be greater than address valid to end of write:

chip enabled = $t_{clav} + t_{dcs}$

data required = $3t_{clcl} + t_{clmh}$

address valid to end of write = t_{aw}

(data required) - (chip enabled) > (addr valid to end of write)

$(3t_{clcl} + t_{clmh}) - (t_{clav} + t_{dcs}) > t_{aw}$

$3t_{clcl} + t_{clmh} - t_{clav} - t_{dcs} > t_{aw}$

Minimize the left side and maximize the right side:

$3(200) + (10) - (110) - (35) > (50)$

$465 > 50$

This condition is satisfied.

3.) Data bus must be tri-stated while Dual Port is still in output mode:

data bus tri-stated by the PC = $t_{clcl} + t_{cvnv}$

bus required to be tri-stated by Dual Port = $t_{clav} + t_{dcs} + t_{as} + t_{wz}$

(bus tri-stated by PC) > (required tri-stated by Dual Port)

$t_{clcl} + t_{cvnv} > t_{clav} + t_{dcs} + t_{as} + t_{wz}$

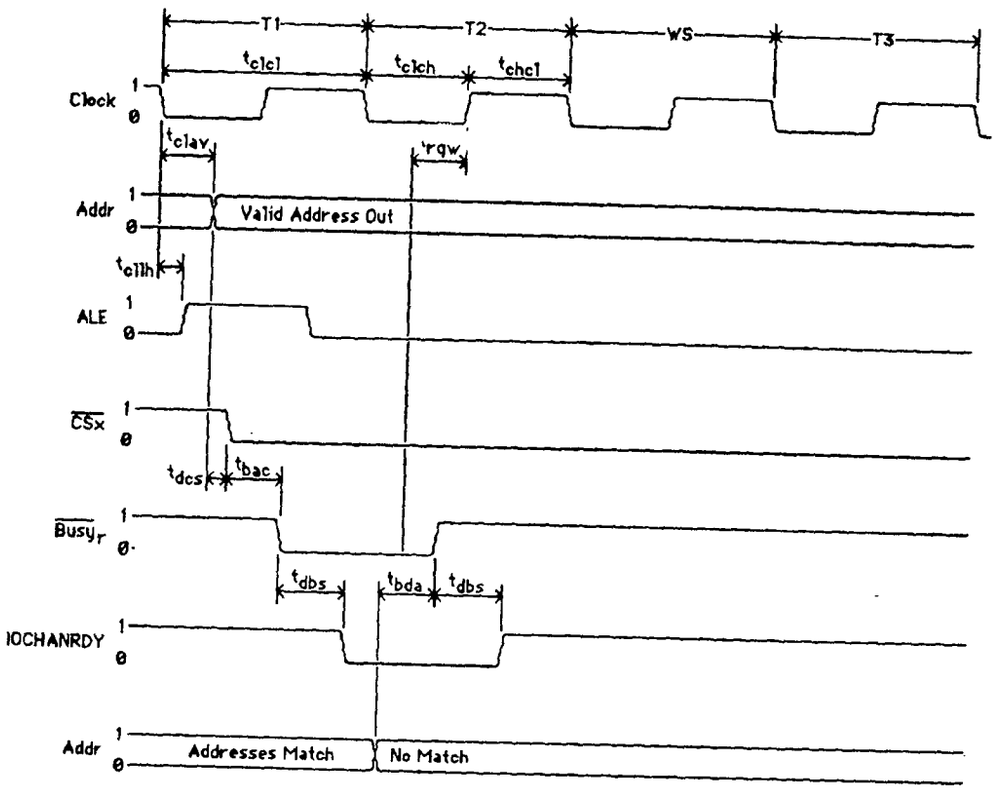
Minimize the left side and maximize the right side:

$(200) + (5) > (110) + (35) + (0) + (35)$

$205 > 180$

This condition is satisfied.

Read or Write Cycle With Busy:



Read or Write Cycle With Busy Calculations:

The following condition must be satisfied for the read or write cycles to correctly generate a busy signal. All numerical values are in nanoseconds.

1.) Valid data must be output by PC before data is read by Dual Port:

busy signal required = $t_{clcl} + t_{clch} - t_{rqw}$

busy signal generated = $t_{clav} + t_{dcs} + t_{bac} + t_{dbs}$

(busy signal required) > (busy signal generated)

$(t_{clcl} + t_{clch} - t_{rqw}) > (t_{clav} + t_{dcs} + t_{bac} + t_{dbs})$

$t_{clcl} + t_{clch} - t_{rqw} > t_{clav} + t_{dcs} + t_{bac} + t_{dbs}$

Minimize the left side and maximize the right side:

(200) + (118) > (110) + (55) + (35) + (45) + (60)

318 > 305

This condition is satisfied.

Timing Diagram Parameters

t_{ace}	Chip enable access time
t_{aoe}	Output enable access time
t_{as}	Address set-up time
t_{aw}	Address valid to end of write
t_{bac}	Busy access time to chip enable
t_{bda}	Busy disable time to address
t_{chcl}	Clock high time
t_{chll}	ALE inactive delay
t_{clav}	Address valid delay
t_{clch}	Clock low time
t_{clcl}	Clock cycle period
t_{cldv}	Data valid delay
t_{cldx}	Data in hold time
t_{cllh}	Clock low to ALE valid
t_{clmh}	Command inactive delay
t_{clml}	Command active delay
t_{cvnv}	Control active delay
t_{dbs}	Delay of busy select logic
t_{dcs}	Delay of chip select logic
t_{dh}	Data hold time
t_{dvcl}	Data in set-up time
t_{dw}	Data valid to end of write
t_{hz}	Output high Z time
t_{lz}	Output low Z time
t_{rqw}	Time before rising clock to request wait state
t_{wp}	Write pulse width
t_{wz}	Write enabled to output in high Z

Part II

The Microcontroller

The microcontroller performs all low level control functions, according to commands passed to it from the PC motherboard. The communication with the PC is done through the dual port RAMs. This hardware is fairly straight forward, but required considerable analysis. The dual port RAM circuit is the major portion of the microcontroller circuitry. Additional connections include the connections to the motor control hardware and encoder logic. The rest of the circuitry was chosen to keep the complexity to a minimum.

The dual port RAM connections are shown in Figure 3.2.1. This is a fairly standard method of accessing memory. The address/data lines are demultiplexed using a pair of 74HC373 latches. Address Valid (ADV) is used to control the latches and Chip Enable (CE, active low) on the RAM. Read (RD, active low) is connected to Output Enable (OE, active low) on the RAM. Microcontroller signal Write WR, active low) is connected to Read not Write (R/W). BUSY (Active low signal from RAM) is connected to READY (active high input on microcontroller).

In the original design, ALE (Address Latch Enable) was used instead of ADV and Chip Enable on the RAM was tied active (low). This design produced a conflict between the PC and the microcontrollers and was changed to the current design. With the old design, after the microcontroller was through accessing an address, the address was left in the latches and Chip enable was still active. This prevented the PC from accessing that memory location until the microcontroller accessed another location. The new design only requires that the microcontroller finish accessing a particular memory location before the PC can access it. Using the Address Valid signal instead of Address Latch Enable also greatly reduces the time that the left port of the RAM is active and therefore greatly reduces its power consumption.

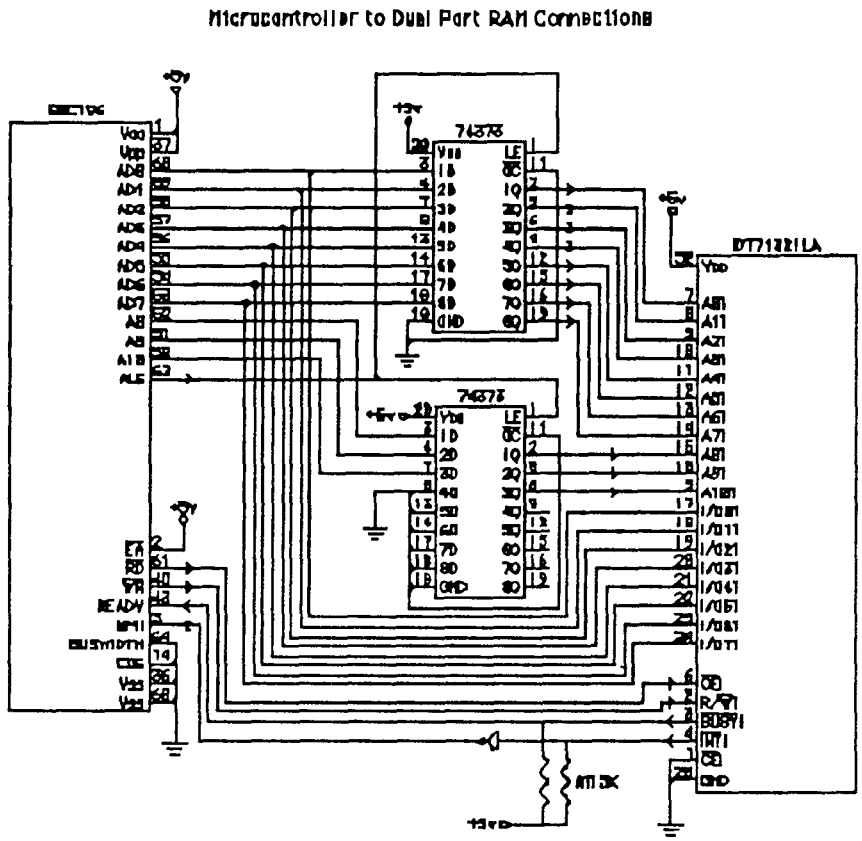


Figure 3.2.1

The RAM is mapped at 4000H to 47FFH in the microcontroller's address space. Since the higher address lines are not used for address decoding, the memory will respond to any location higher than 4000H. These "shadows" of the memory should not be used, as they are reserved for later use.

The memory design allows extremely fast and versatile communication between the PC and each microcontroller. Both have full simultaneous access to the same physical memory block. This allows many different data transfer schemes to be used. To pass commands that require immediate execution, an interrupt feature of the dual port RAMs will be used. To use this feature, one CPU will store data in a certain memory location (a "mailbox"). This will generate an interrupt signal for the other CPU. The second CPU will service the interrupt and clear the signal by reading data from its "mailbox". There are two mailbox locations, 47FEH and

47FFH, one for passing commands each direction. The PC will write to location 47FEH to cause an interrupt on the microcontroller. This interrupt line on the microcontroller side is connected to the non maskable interrupt.

The major consideration for this memory design is how fast will it function correctly. Both the memory and the microcontroller give timing specifications as to how they will perform. The 80C196KB12 and 80C196KB10 microcontrollers have slightly different timing specifications even if running at the same frequency. Timings were calculated for both versions. It was necessary to analyze each of the requirements given by the microcontroller to determine whether the system will respond within the correct time frame in all cases. These calculations are included in table 2.1.

With the completion of the timing calculations on both the microcontroller side and the PC side, it was determined that the system will work correctly at 10 MHz or slower with a 55 ns (or faster) memory. The 55 ns memory is the desired one because it is the slowest non-military version available, and therefore the least expensive. In order not to push the capabilities of the system, it was decided to run the system at a slower rate. It is not necessary to run the microcontroller at a high speed because the speed is only needed in the I/O to the motor control circuits. These circuits are inherently fast because they are connected to the high speed inputs and outputs (HSIO port) and the pulse width modulator (PWM). This high speed I/O port can run almost as fast as the microcontroller can pass data to and from it. Thus the I/O frequency can easily be 1/100th of the clock frequency. (The PWM is even faster.) A speed of 3.5 MHz (the slowest allowable for the 80C196) will allow at least one access every .0003 seconds, which is more than fast enough for the motor driver circuits.

The PC bus has its oscillator line (OSC) available on the PC bus, and this is used to drive a frequency divider circuit, which will drive the clock frequency input on the microcontroller, XTAL1. XTAL2 is floated when using an external clock drive. CLOCKOUT is not connected simply because no external circuitry uses it. The OSC line from the PC has a frequency of 14.3 MHz. A divide by two circuit will cause the microcontrollers to run at 7.15 MHz, which is in the desired range. This circuit requires that the OSC line conforms to the "External Clock Drive Waveform" specifications given in the data sheet. Clock Detect Enable (CDE) is grounded because Intel does not guarantee the clock fault detect circuit to work correctly, and

the circuit may inadvertently reset the microcontroller if enabled.

The RESET circuit is a standard RC circuit which charges whenever the power is on and drains whenever the power is off. The output is run through a Shottky inverter to provide sharp transitions and buffer the signal.

The external I/O connections are shown in Figure 3.2.2. These connections use the high speed input and output lines for most of the motor drive circuit. The PWM line is used for the walking motor because it is easier to program than the high speed output and takes less CPU time. The high speed output is used for the PWM on the lifting motor. The high speed input is used to receive data from the encoder circuitry. The various switches are individually connected to bits of port 2, and collectively (through an OR gate) to the external interrupt pin (EXTINT) so that an interrupt routine may be used to service the switches.

80C196 Pin Name,	Number,	Type	External Device	Line
PWM	39	Output	Main Drive Motor	PWM
P2.6	45	Output		Direction
P2.1	61	Input		Limit Switch
HSI.0	54	Input	Encoder	Direction
HSI.1	53	Input		Count
HSO.0	50	Output	Lifting motor	PWM
P2.7	40	Output		Direction
P0.0	4	Input		Limit Switch
HSI.2	52	Input	Encoder	Direction
HSI.3	51	Input		Count
P2.4	36	Input	Foot Contact Switch	

Figure 3.2.2 I/O connections

Several other connections are required for proper operation of the microcontroller. P0.0 through P0.7 are used for a "debugging port"; they are connected to two four-bit hexadecimal displays. T2CLK is connected to a line of the OSC divider circuit to run at 1/16th the speed of XTAL1. Several connections to the A/D converter are needed even if it is not being used. The Voltage reference (V_{ref}) and ground (ANGND) must be connected (see Figure 3.2.3). Analog ground and digital ground are connected at the power supply. The two V_{SS} pins were directly connected to prevent a voltage difference between them.

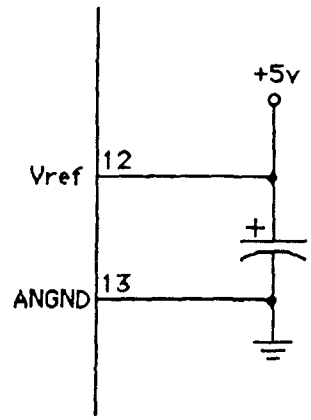


Figure 3.2.3 Analog to Digital References

System Timings
Microcontroller to Dual Port RAM

The following timings are required by the microcontrollers or the system will not function properly. This table is condensed from INTEL's data sheet.

Name	Min (ns)	Max (ns)
T_{AVYV}		81/115
T_{LLYV}		11/20
T_{CLYX}	0	
T_{LLYX}	68/85	
T_{AVGV}		N/A
T_{LLGV}		N/A
T_{CLGX}		N/A
T_{AVDV}		182/230
T_{RLDV}		60/70
T_{CLDV}		33
T_{RHDZ}		63
T_{RXDX}	0	

Each of these timings is calculated in the following table. With the exception of the three timings discussed here, all timings are met by this design.

The "Ready hold after CLOCKOUT low" time (T_{CLYX}) and "READY hold after ALE low" time (T_{LLYX}) are required to insure that a wait state is inserted. These calculations were made with the assumption that not meeting these minimum READY hold times will cause nothing worse than not having a wait state. To confirm this an engineer at Intel was consulted. He confirmed that there would be no adverse effects from not meeting the minimum READY hold time. There are cases when these timing requirements are not met. Further calculations showed that in these cases, the RAM will respond with the data fast enough in all cases except with the 55 ns or 70 ns memory and the microcontrollers running at 12 MHz.

The "Address valid to READY setup" time (T_{AVYV}) will not be met while running at 12 MHz even with the faster memory.

Notation:

t_{XXX} Small case t indicates time defined by the RAM

T_{XXXX} Capital T indicates time defined by the controller

12/10 2 numbers

- First time is for an 80C196KB12 at 12 MHz
- Second time is for an 80C196KB10 at 10 MHz

35/45/55/70

4 numbers

- one for each available speed of the RAM

35/45/55/70-35/45/55/70

8 numbers

- Four at 12 MHz
- Four at 10 MHz

Omitted numbers are the same as the previous number.

i.e. 35//45/ = 35/35/45/45

All times are in nanoseconds.

The names used here are the same as those used in the data sheet.

<u>ACTIVE LOW</u> signal names are underlined.

The following are the calculations used to determine if the system will satisfy the necessary timings for various speed/part combinations:

1) Address valid to Ready valid time (T_{AVYV}): max allowed = 81/115

$T_{AVYV} = [\, T_{LHLL}\,(max) - T_{AVLL}\,(min)\,] + $ '393 delay + t_{BAA}
$= 25 + 30 + 35//45/$
$= 90//100/$

This is the critical timing: Address valid to READY valid. None of the RAMs can respond fast enough to insure a wait state if the controller is running at 12 MHz.

2) ALE low to READY setup Time (T_{LLYV}): max allowed = 11/20

$T_{LLYV} = -[\, T_{LHLL}$ or $T_{AVLL}\,] + $ '373 delay + t_{BAA}
$= -66 + 30 + 35//45/$
$= -1//9/$

3) READY hold after CLOCKOUT low (T_{CLYX}): min allowed = 0

4) READY hold after ALE low (T_{LLYX}): min allowed = 68/85

T_{CLYX} and T_{LLYX} cannot be guaranteed to be satisfied. If the max is exceeded, an extra wait state will be added. Extra wait states are not a problem in this design. If the min is not satisfied, no wait state will be generated, and the response of the RAM needs to be fast enough to correctly store or retrieve the data. The read and write cases will be analyzed separately.

T_{CLYX} (read cycle)
$T_{RLDV} = 60/70$
t_{BDD} = max of 0, 30/35/40/, 15/25/35/40 = 30/35/40/
$T_{RLDV} >= t_{BDD}$ So in this case no wait state is needed.
RD won't go low before READY goes high.

T_{LLYX} (read cycle)

required response = $T_{LLRL} + T_{RLDV}$ = 43/60 + 60/70 = 103/130

actual response = $T_{LLYX} + t_{BDD}$ = 68/85 + 30/35/40/

= 98/103/108/108 - 115/120/125/125

The response is not fast enough to guarantee a correct read when no wait state is inserted is two cases: 55 ns RAM at 12 MHz and 70 ns RAM at 12 MHz.

T_{CLYX} (write cycle)

If no wait state is generated, the RAM needs the data to be held on the bus for long enough to store it.

RAM needs (max allowed):

$T_{LLCH} + T_{CHCL} + t_{WH}$ = 15 + 93/110 + 20 = 128/145

RAM gets (actual response):

$T_{LLWL} + T_{WLWH} + T_{WHQX}$ = 73/90 + 53/70 + 73/90 = 199/250

T_{LLYX} (write cycle)

RAM needs (max allowed):

$T_{LLYX} + t_{WH}$ = 68/85 + 20 = 88/105

RAM gets (actual response):

$T_{LLWH} + T_{WLWH} + T_{WHQX}$ = 73/90 + 53/70 + 73/90 = 199/250

5) Address valid to input data valid (T_{AVDV}): max allowed = 182/230

T_{AVDV} = [T_{LHLL} (max) - T_{AVLL} (min)] + '373 delay + t_{AA}

= 25 + 30 + 35/45/55/70 = 90/100/110/125

6) Read active to input data valid (T_{RLDV}): max allowed = 60/70

$T_{RLDV} = t_{AOE}$ = 25/30/35/40

7) CLOCKOUT low to input data valid (T_{CLDV}): max allowed = 33

This cannot be calculated directly, so the time from latch low to input data valid will be calculated (using T_{CLDV}). The memory must respond in less time than the response required by the controller.

T_{CLDV}

Response required by controller:

T_{LLCH} (min) + T_{CHCL} (min) + T_{CLDV} = -15 + 73/90 + 33
= 91/108

Actual memory response:

T_{LLRL} + t_{AOE} = 43/60 + 25/30/35/40
= 68/73/78/83 - 85/90/95/100

8) End of read to input data float (T_{RHDZ}): max allowed = 63

T_{RHDZ} = t_{HZ} = 15/20/30/35

9) Data hold after read inactive (T_{RXDX}): min allowed = 0

This requirement simply specifies that the data must be kept on the data lines until after the read signal goes inactive. This satisfied by the design of the control lines.

Timing Diagram Parameters

T_{AVDV}	Address valid to data input valid
T_{AVGV}	Address valid to Buswidth setup
T_{AVLL}	Address valid to ALE falling edge
T_{AVYV}	Address valid to READY setup
T_{CHCL}	CLOCKOUT high period
T_{CLDV}	CLOCKOUT low to data input valid
T_{CLGX}	Buswidth hold after CLOCKOUT low
T_{CLYX}	READY hold after CLOCKOUT low
T_{LHLL}	ALE high period
T_{LLCH}	ALE falling edge to CLOCKOUT rising edge
T_{LLGV}	ALE low to Buswidth setup

T_{LLRL}	ALE falling edge to READ falling edge
T_{LLYV}	ALE low to READY setup
T_{LLYX}	READY hold after ALE low
T_{RHDZ}	End of READ to data input float
T_{RLDV}	READ active to data input valid
T_{RXDX}	Data hold after READ inactive
T_{WHQX}	Data hold after WRITE rising edge
T_{WLWH}	WRITE low period
t_{AA}	Address access time
t_{AOE}	Output enable access time
t_{BAA}	BUSY disable time to address
t_{BDD}	BUSY disable to valid data
t_{HZ}	Output high Z time
t_{WH}	WRITE hold after busy

Part III
Real-World Interface

The electrically noisy environment of the motors is kept separate from the sensitive computer circuitry by using opto-isolators. The power for the motors is two 12 volt batteries connected in series. The computer circuitry uses a separate battery to isolate the computer circuitry further.

Motor Circuitry

Each motor is hooked up through a relay to reverse the motor in the simplest way possible (Figure 3.3.1). To prevent arcing of the relay contacts, the motors must be stopped before changing their direction.

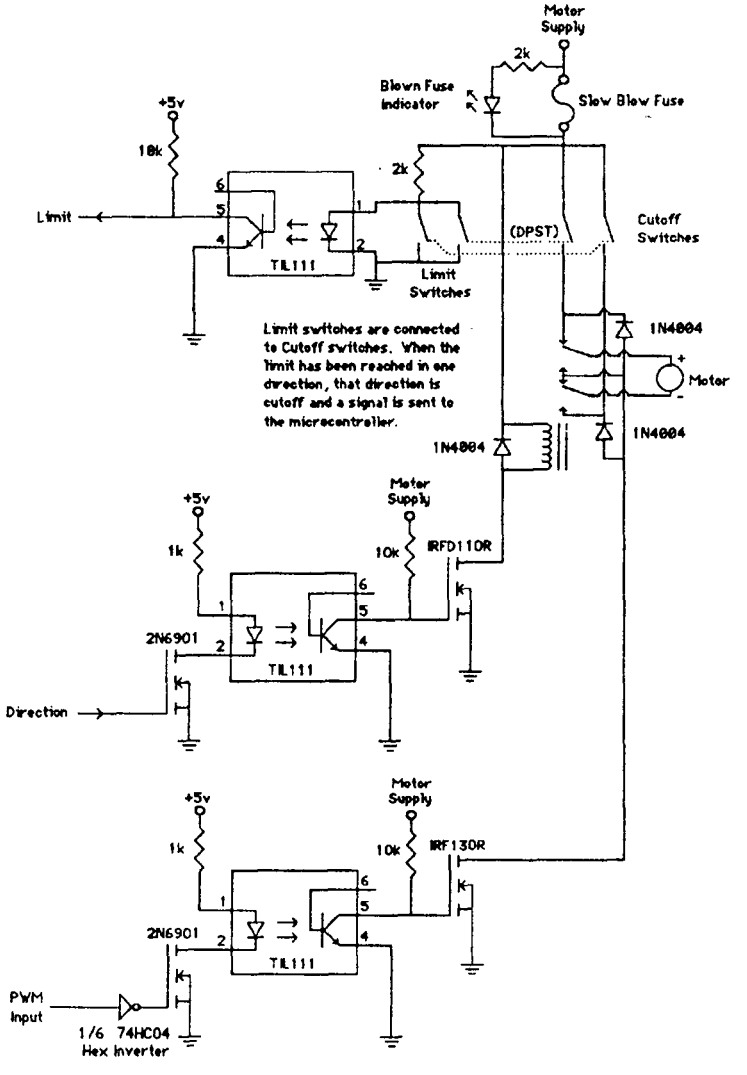


Figure 3.3.1: Motor Control Circuitry

Motor speed is controlled by using pulse width modulation (PWM). PWM provides a motor with high peak current, but lower average voltage. The transistor for PWM is rated at a maximum continuous current of over 10A, with much higher surge currents. The fuse prevents any continuous currents higher than designed for. The LED across the fuse will be lit only when the fuse is blown. This provides an immediate visual indication of a problem with any of the fuses.

Hardware limit switches provide a fail-safe mechanism to stop the motors if the computer fails to turn a motor off at the proper time. Given the power and gearing of the motors, this is necessary; without it, the robot has the potential of damaging itself.

Encoder Circuitry

Figure 3.3.2 is an optical encoder decoder schematic. It reads an encoder and converts the data from the encoder into a 'Count' and a 'Direction' signal. 'Count' is a wave that gives an incremental indication of the rotation of the motor. 'Direction' is high or low, depending on the direction of rotation of the motor.

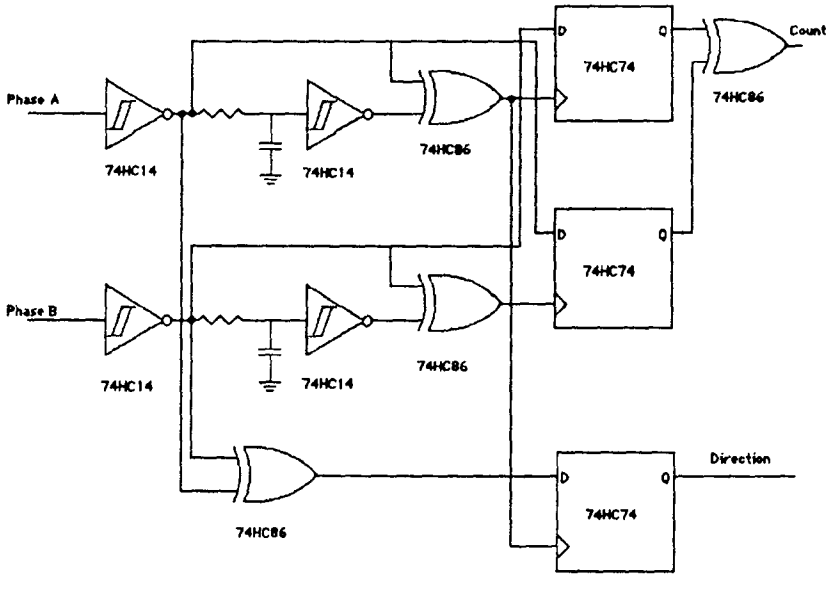


Figure 3.3.2: Encoder Circuitry (Schaefer, p. 4)

The resistor and capacitor values are dependent upon the frequency of the encoder pulses. The values must be computed to give a delay that is smaller than a pulse directly off of the encoder, but larger than noise that would cause the circuit to operate erratically. They were determined experimentally with the working hardware.

Voice Recognition Circuitry

Voice control is necessary in certain situations. This was done by using a commonly available chip that costs about $10.00 and recognizes 5 separate commands. It is based on a speaker independent voice recognition algorithm. This chip and a few external parts will provide 5 commands: GO, TURN RIGHT, LEFT TURN, REVERSE, STOP.

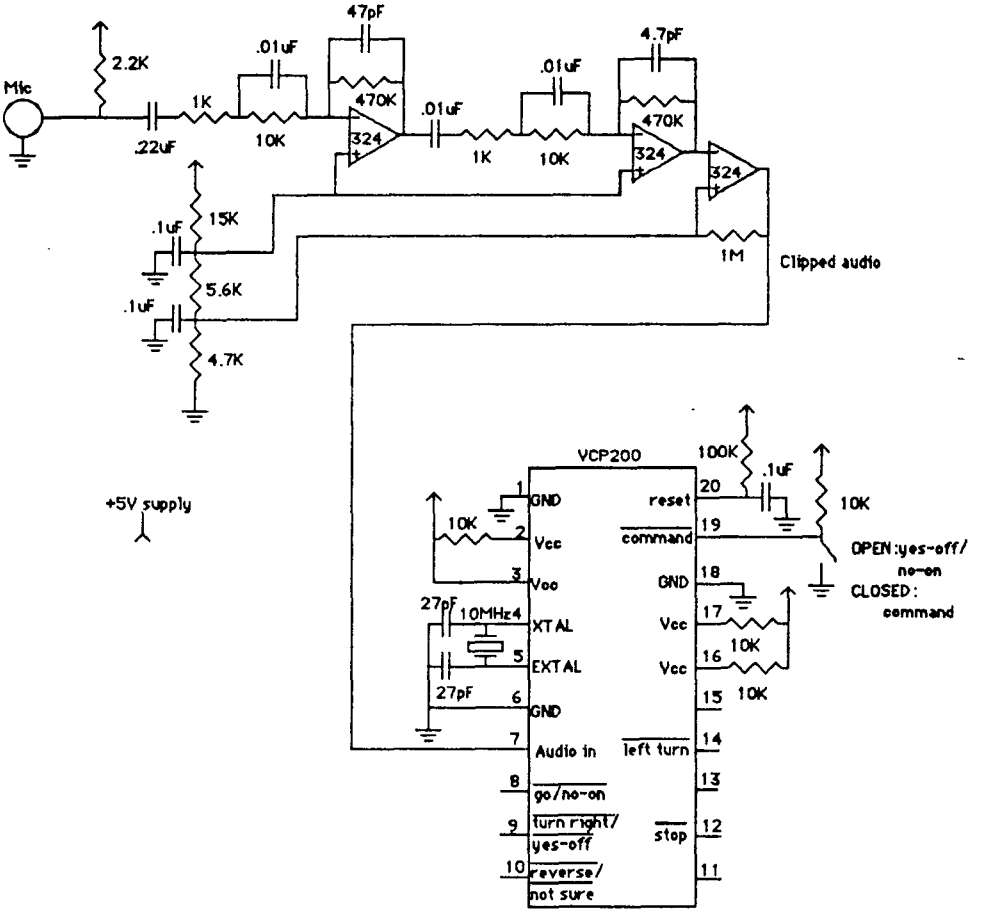


Figure 3.3.3: Voice Recognition Circuitry (Archer, p.3)

Power Supply

The computer's power-supply must be heavily regulated. A digital circuit is sensitive to noise on it's supply voltage. If the power is not regulated sufficiently, the computer will not operate properly or consistently. Here, there are two options, a series linear regulator, or a switching power supply. The series is simpler, but the switching power supply is much more efficient. (A significant concern, since the robot is battery operated.)

Foot Sensing Switch

The robot must have some way of sensing that it's foot has contacted the ground. A simple opto-isolator circuit is used to reduce the risk of damaging the microcontroller hardware. Figure 3.3.4 is a circuit that provides an active high indication of the foot contacting the ground. It is a general purpose circuit that can be used whenever a switch needs to be interfaced to the computer hardware.

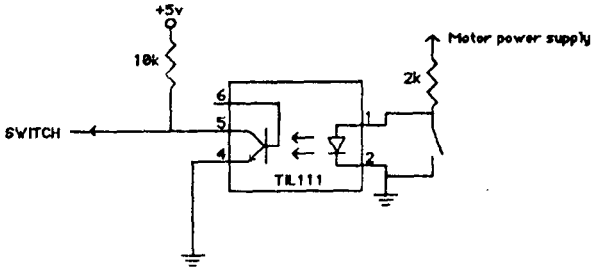
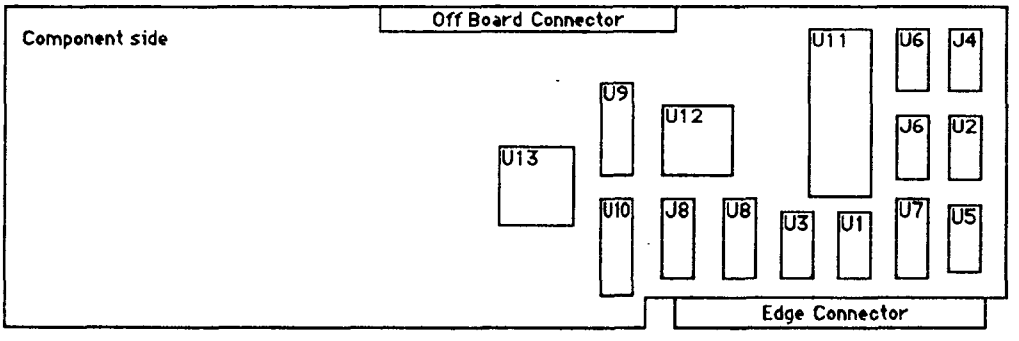


Figure 3.3.4: Foot Sensor switch

Part IV
Construction Notes

PC interface

The PC interface hardware was constructed on an expansion card. The hardware was wire wrapped on the perf board card. Figure 3.4.1 shows the layout of the PC interface card. This expansion card was used to allow construction of the hardware while the card was removed from the PC. The card could be remove to be worked on and then reinstalled in the PC for testing. The card was constructed using a color coded wiring system. This was done to ease debugging, particularly to help find mistakes from incorrect wiring. The layout allows for expansion, since only about half of the card is used.



U1 : 74HC4002
U2 : 74HC04
U3 : 74HC00
U5 : 74HC30
U6 : 74HC30
U7 : 74HC138
U8 : 74HC138
U9 : 74HC373
U10 : 74HC373
U11 : 8255
U12 : IDT71321
U13 : 80C196
J4 : Busy jumper
J6 : Interrupt Request select
J8 : PPI Address select

Figure 3.4.1

This hardware could be modified to ease construction. The circuits could be redesigned using programmable array logic (PAL) chips. This would reduce the address decoding to two chips, one for the memory selection and one for the PPI selection. Two PALs could replace five chips and a jumper block. This would save space and debugging time. It would also allow hardware modifications by replace the PALs with differently programmed ones. From a production standpoint, this entire board could be manufactured on a single medium scale integration (MSI) chip. This would reduce required space, simplify the connections, and simplify debugging. To reduce space even further, the entire PC and PC interface hardware

could be put on a large or very large scale integration (LSI or VLSI) chip.

Microcontroller Board

Due to space considerations the leg microcontrollers were placed on a separate 8.5" by 17" perf-board. A common bus was placed along one edge of the board to distribute the data and address busses from the PC interface card to the microcontrollers. The dual-port memories were placed closest to the bus. The opto-isolators were placed on the opposite side of the board to isolate the power circuitry from the computing circuitry. The microcontrollers were placed in between the opto-isolators and the dual-port memories.

The microcontroller board has a huge number of interconnections. The wire-wrapping techniques used do not adapt well to the environment on a walking robot. By mounting the microcontroller hardware on printed circuit boards, the microcontrollers would be much more reliable and durable. This is quite a bit more expensive. The boards must be thoroughly debugged before this should be done.

Power Supply

The robot is battery powered so a switching supply was desired to conserve battery life. To determine the supply requirements it was necessary to finalize the power needs of the robot's five volt circuitry. The bulk of the supplied load is from the PC motherboard and disk drive. After testing constructed systems and estimating the needs of proposed circuitry, a maximum four amp, five volt load was established.

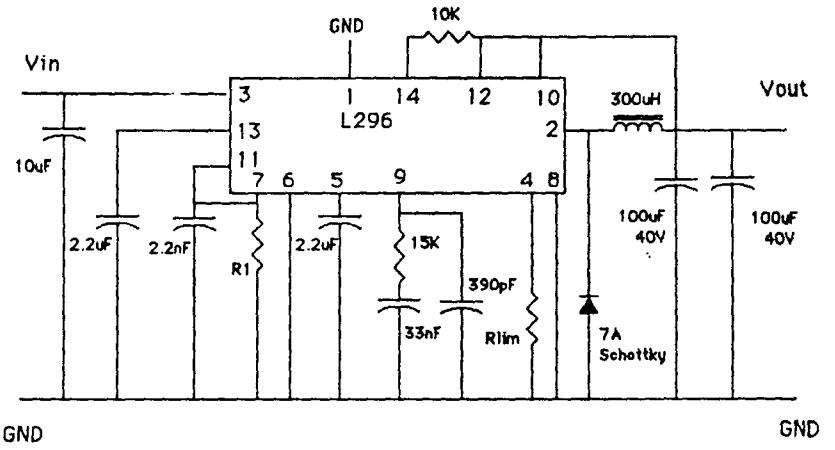


Figure 3.4.2: L296 High Current Buck Regulator

Once the requirements had been finalized, existing switching power supplies were reviewed in search of a five volt high current supply. The final decision was

an L296 High Current Buck Regulator, a 5 to 40 volt 6 amp regulator, from the Unitrode;. The databook also provided a common application of this IC that proved to be the basis of the design. The circuit is shown in Figure 3.4.2. R_1 was set to 4.3K to yield a switching frequency of 160 kHz and a resulting 74% efficiency.

After designing the circuit board layout, transfers were used to mask the copper board prior to etching. This circuit could not be breadboarded because of high current outputs and was therefore tested after construction. Initially the results were disturbing as the supply produced a constant 5 volt output for low current loads (200mA); but when the load was decreased to test the available current range the output dropped to 0.8 volt for a load of 300mA or greater. Troubleshooting this circuit became a formidable task given the limited information supplied by Unitrode. After several tests the solution was to remove the current limiting resistor R_{lim}, which according to the databook should default the current limit to 6A. A 5A fuse was placed in series with the supply's output to protect the supply and other circuitry in the event of a short circuited load.

Once these difficulties where alleviated the supply circuit was tested using a 12 volt battery and artificial loads. Successful completion of these tests allowed the direct connection of the supply into the robot's circuitry. The final implementation of this circuit in the robot performed without difficulties, meeting the design goals and current requirements.

Motor Control and Relay Boards

The computer circuitry output signals, used for controlling the thirteen motors used in this robot's design, are of an insufficiently high-enough voltage to drive the 24 volt motors used, and of too sensitive a nature to be directly connected to the high voltage motor-side of this system. There must be some circuitry which accepts these low voltage controlling signals and translates them into high voltage signals capable of driving the turning mechanism's and legs' motors, and some means of intermediate protection between the two sides of this system.

The sensitive computer circuitry and the relatively noisy environment of the robot's motors are kept separate through the use of opto-isolators. Two transistors per motor are present on the motor-side of this system; a Pulse Width Modulation MOSFET, which controls the speed of the motors, and a transistor which determines direction. Each of these transistors has its gate connected to the intermediate opto-isolators and are "tied-high" to 24V by means of a 10K resistor. A fuse and "blown-

fuse" indicator LED are included in series with each motor, as a form of protection against large current surges and as an indication of such surges respectively.

In addition, each motor is connected through a relay which acts to reverse the motor direction. It was decided to mount the relays on two separate boards using barrier blocks for connections. This was done to ease trouble-shooting and simplify the re-wiring that would be done to "fine-tune" the operation of the robot.

Since the two relay boards were simple in design, it was decided to use point-to-point soldering for their construction. A decision was made to place all motor inputs on one side of the board and all motor outputs on the opposite side. Twenty gauge, solid-core wire was used in the construction because of the currents involved (limited by fuses to 5A). This is done to decrease the time it takes the relay contacts to go from the normally-closed position to the normally-open switch position, called the "pull-in" time, and to reduce the relay coils tendency to produce electrical noise in the circuit, known as "backwards EMF".

Because the motor-side of this system is a high current environment, a decision was made to mount all motor-side components on etched circuit boards with more than sufficiently wide copper traces for the current they would carry. An etched circuit board was deemed preferable to soldering heavy gauge wire on a point-to-point basis for a number of reasons: (1) Point-to-point soldering of 7 components per motor is a time consuming task and is difficult to keep orderly, (2) trouble-shooting is more easily accomplished, and (3) fast and easy replacement of defective or ruined components is aided.

The motor control circuit board design was created with the aid of a printed circuit board design software package. The final foil pattern design was as shown in Figure 3.4.3.

After construction, these boards were tested and were found to work correctly in all respects, requiring no debugging or modifications.

Voice Recognition Circuitry

During construction of the robot's electrical hardware systems, strong emphasis was placed on completing construction of those systems designated as essential to having the robot perform its most basic tasks: walking and turning. Only after it was clear that all of the major systems were near completion was someone assigned to the construction of non-essential circuits. Included in this group of circuits was the voice recognition circuitry.

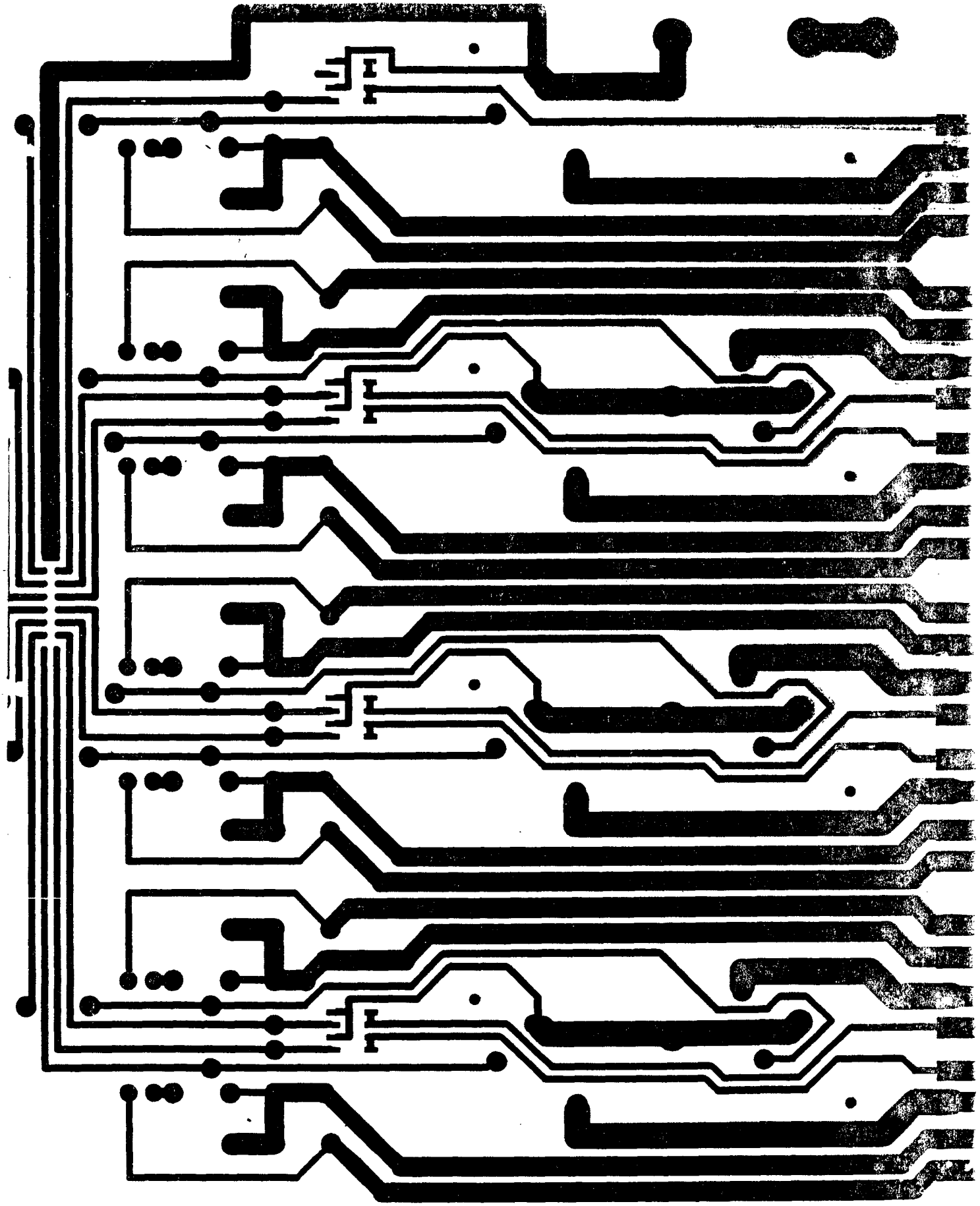


Figure 3.4.3: Motor Control Board Foil Pattern

The voice recognition circuitry was constructed around the VCP200 Speaker-Independent Word Recognizer. In its command mode, this chip recognizes 8 commands of which the five most important movement commands are GO, STOP, LEFT TURN, TURN RIGHT, and REVERSE. The outputs are active low.

The VCP200 data sheet includes foil patterns for printed circuit board implementation. However, as these patterns are for a two-sided board, an alternative single-sided foil pattern, presented in a Radio Electronics magazine article about this same chip, was used for greater ease of construction. Changes made to the layout included the elimination of the power circuitry, as a 5 volt power regulator was included in the design of the robot, and instead of the suggested microphone, and electret condenser microphone was used to improve the circuit's input signal in noisy environments. Also, a switch was included to disconnect the voice recognition circuitry from the external power supply.

After construction, this board was tested and was found to work correctly in all respects, requiring no debugging of the circuitry.

Encoder Boards

As discussed in the design , the encoder circuitry was taken directly from Intel's Application Notes. There were, however, some remaining requirements of the design to be finalized before a prototype could be constructed. The circuitry utilizes a series of Schmitt triggers, exclusive OR gates, and delay flip-flops to monitor motor speed and direction.

The encoder initially levels off the input signals then passes them through delay filters. The delay is necessary to make a comparison with the originally unaltered signal in the counting process. The actual value of the RC-network had not yet been determined; thus the objective was to determine the necessary delay in the circuit and fix component values to accomplish this goal. To initiate the design, the capacitor was fixed at 0.1 uF, and the expected wave forms through-out the circuit were plotted. Knowing the maximum expected input frequency (250 Hz), the delay time was set at one sixth of the maximum input period (delay=0.17mS). One sixth was selected to limit any possible error that could occur when the motor changes direction during. Using the transfer expression for the RC-network a resistance of 1K was used.

The board was laid out to take advantage of the output pin symmetry and shared components among the different integrated circuit chips then sectioned to

separate the circuits into the different leg groups with two motors per leg sharing the same components and input sockets. Then the leg circuits were split into two similar boards with each board assigned to three legs apiece and the circuit for the turning motor added to one of the boards.This is shown in Figure 3.4.4. A flow-

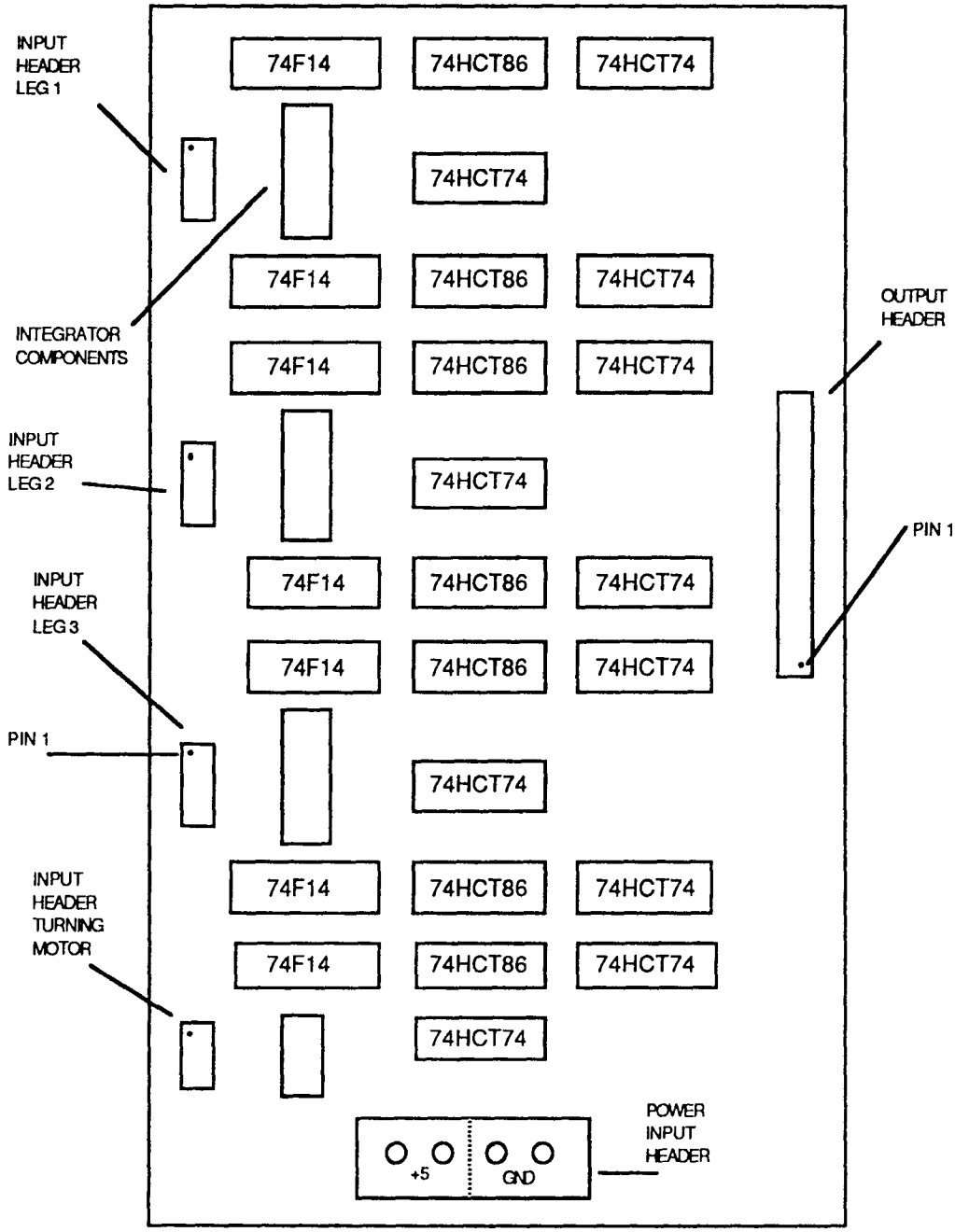


Figure 3.4.4: Encoder board layout

through architecture, with the input signal directed into one side of the board and the output taken from the opposite side, was selected to avoid cabling problems.

For the sake of simplicity, the construction consisted of perforated board and wire-wrap IC sockets. Sixteen pin sockets were chosen to easily accommodate the decoupling capacitors without soldering. An IC socket was also used for the resistor capacitor integrator to provide for a quick change of components and to speed up construction.

The boards were tested by connecting a motor encoder to the circuits and observing the outputs while the motor speed and direction were varied. Adjustments to the integrator were made by changing the component values of the resistor or capacitor to maximize the efficiency of the circuit.

Due to the efficiency of the layout and circuit, the only improvement of the motor encoder board would be a small reduction in space and current gained by utilizing the unused portions of the chips.

Circuit Board Etching

When it had been decided that etched circuit boards would be used in those robot systems that could benefit from such an implementation, discussion was held as to whether those circuit boards should be contracted out to a vendor, or if they should be produced by our own group. After making a number of calls to local vendors, it was determined that the cost of having the necessary boards produced by an outside source would be exorbitant.

At this point, means of etching circuit boards "in-house" were discussed. Two methods to pursue were agreed upon:

(1) Photo-resist etching. In this process, the copper-clad circuit board to be used was sprayed with a photosensitive material in the absence of light and allowed to dry overnight. The foil pattern to be etched was reversed, black-for-white, and photo-copied onto a transparency. When the board was completely dry, the transparency was fixed to the copper board. The board was then exposed for a set time to a strong ultra-violet (U.V.) light source, and where the U.V. light struck the photosensitive material, that material was sensitized. The board was then developed in an appropriate developing solution, with the sensitized material hardening and turning opaque. At this point, the board was placed into an etching solution of ferric chloride where all exposed copper was to be removed. Unfortunately, due to the unavailability of a strong enough U.V. light source, we were unsuccessful in etching the boards by this process.

(2) The "toner" method. This method makes use of a photo-copy transparency of

the circuit foil pattern and a common household iron. The foil pattern to be etched is reversed, left-to-right, and photo-copied onto a transparency, being careful to have heavy toner deposited on the plastic during photo-copying. This transparency is then laid on top of the copper-clad board, toner-side down, and ironed with a hot iron until the majority of the toner is deposited on the copper. Afterwards, a permanent marker was used to touch-up those areas not well transferred. Each board was then etched in a ferric chloride solution were all exposed copper was removed. This simple method had surprisingly remarkable results and provided an inexpensive, quick, and reproducible method for etching small circuit boards.

The primary design goal of PredaTerp's software team was to implement an optimal solution to controlling the complex mechanical and electrical systems. The simplest implementation of the software would occur on a single processor computer architecture. The break down would look like Figure 4.1.

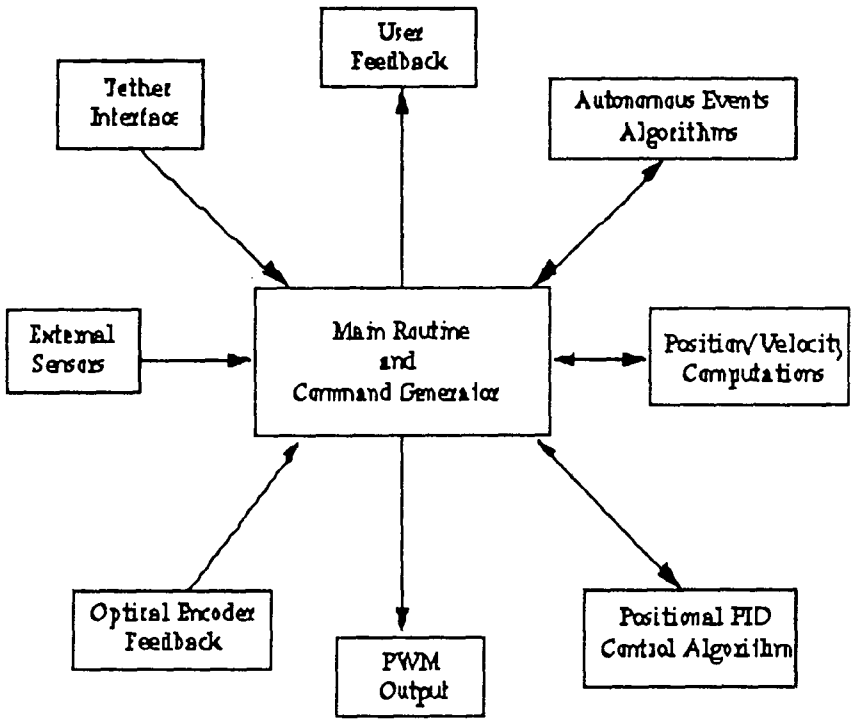


Figure 4.1

The *Main Routine and Command Generator* would be responsible for the initialization of the software and for coordination of the different software tasks. The *Tether Interface* routines allows for a human operator, who would have the flexibility to either control PredaTerp manually, or invoke one of several *Autonomous Event Algorithms*, which, with the help of the *External Sensors* interfaces, allow PredaTerp to operate without human guidance. *User Feedback* is provided for ease of operation. For PredaTerp to be able to complete any task, it is necessary to have *Optical Encoder Feedback* from the various motors, to allow for *Positional Proportional/Integral/Differential (PID) Control. Pulse Width Modulation (PWM) Output* drives the motors.

Implementing this software design would not do justice to the flexibility available in PredaTerp's computer hardware. As stated in the Hardware section,

PredaTerp utilizes a multiprocessor system, where an INTEL 8088/6, in the form of an IBM PC clone mother board, is used as the "coordinating" processor, and seven INTEL 80C196 microcontrollers are used to control the various motors. In order to utilize the strengths of the various components of this hardware system, the software tasks had to be split. Obviously, one would use the 80C196s to run the code that would control the motors, a task that the PC is not suited for. Many of the other tasks could be allocated to either the PC or the 80C196s. The determining factor was the interprocessor communication bottleneck.

It became apparent that serious timing problems could result if too much was assigned to the PC. The PC would be faster and more accurate at doing the various calculations, especially the Positional Control Algorithms. However, doing that calculation for as many as thirteen motors at once would be a logistical nightmare. A similar problem arises if too much is assigned to the 80C196s. The 80C196 is a very capable microchip, but lacks the computational prowess of the 8086/8. In addition, a lot of potential operational speed of the 80C196s was sacrificed when it was decided to limit the clock speed of the 80C196s to that of the operational clock speed of the PC in order to simplify the hardware design.

The PC was to act as the supervisor/coordinator, issuing commands to the microcontrollers. The microcontrollers were to actually operate the motors in a manner that would fulfill the commands of the PC, and provide some feedback to the PC. So, inspite of the flexibility afforded in the use of the Dual Port RAMs (DP-RAMs), the DP-RAMs were to merely pass simple commands and feedback back and forth. (This is not entirely true, as is to be shown later). The breakdown of the software tasks can be seen on the following pages in Figure 4.2 and Figure 4.3.

The PC was assigned tasks that allowed it to easily fulfill its supervisory role. The *Tether Interface, User Feedback,* and *Autonomous Events Algorithms* are run on the PC so that either the operator or PredaTerp can decide what tasks need to be completed in order to accomplish the goal. The *PC Main Program and Command Generator* breaks down these tasks into commands that are issued to the 80C916s. Control over the *Environmental Sensors* was given to the PC in order to allow for ease of autonomous operation. The PC is also responsible for self initialization (*Main PC Initialization*) and for initialization of the complete PredaTerp hardware platform (*System Initialization*). The various 80C196's commands are written out to the appropriate DP-RAMs (*Write to Dual Port RAM's*), and, upon receipt of the

appropriate interrupt (*Receive Mailbox Interrupt*), the PC poll the various DP-RAMs to receive feedback from the 80C196's (*Read Dual Port RAMs*).

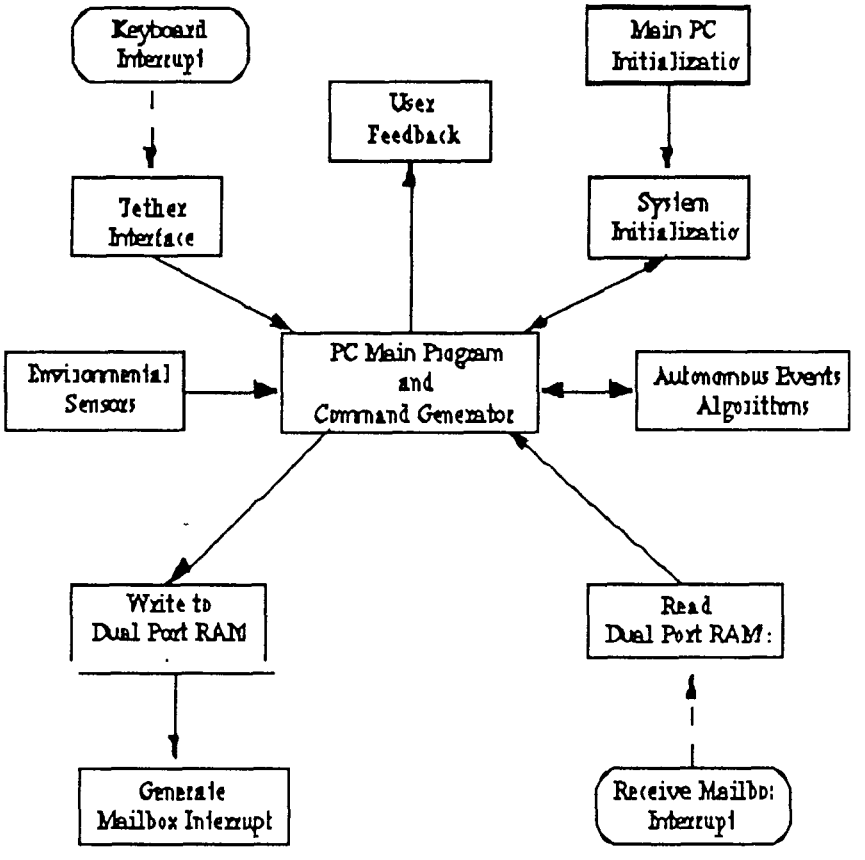


Figure 4.2

The 80C196s, which, aside from the *Intel 80C196 Initialization Routine,* is completely interrupt driven, have all the routines necessary to control the motors. The microcontroller receives commands from the PC via the *Read Dual Port RAM* routine when a *Receive Mailbox Interrupt* is actuated. Based on the command, the *PID Control Algorithm* calculates the next desired position of the motor. *Pulse Width Modulation Output* is provided to control the motors, and motor *Position and Velocity Calculation* feedback is provided via the *Optical Encoder Interrupt.* The microcontroller is able to *Reset Motor Position Count* when the leg or body reaches its limit of travel and the *Motor Limit Switch Interrupt* occurs. Feedback to the PC is provided via the *Write to Dual Port RAM* routine.

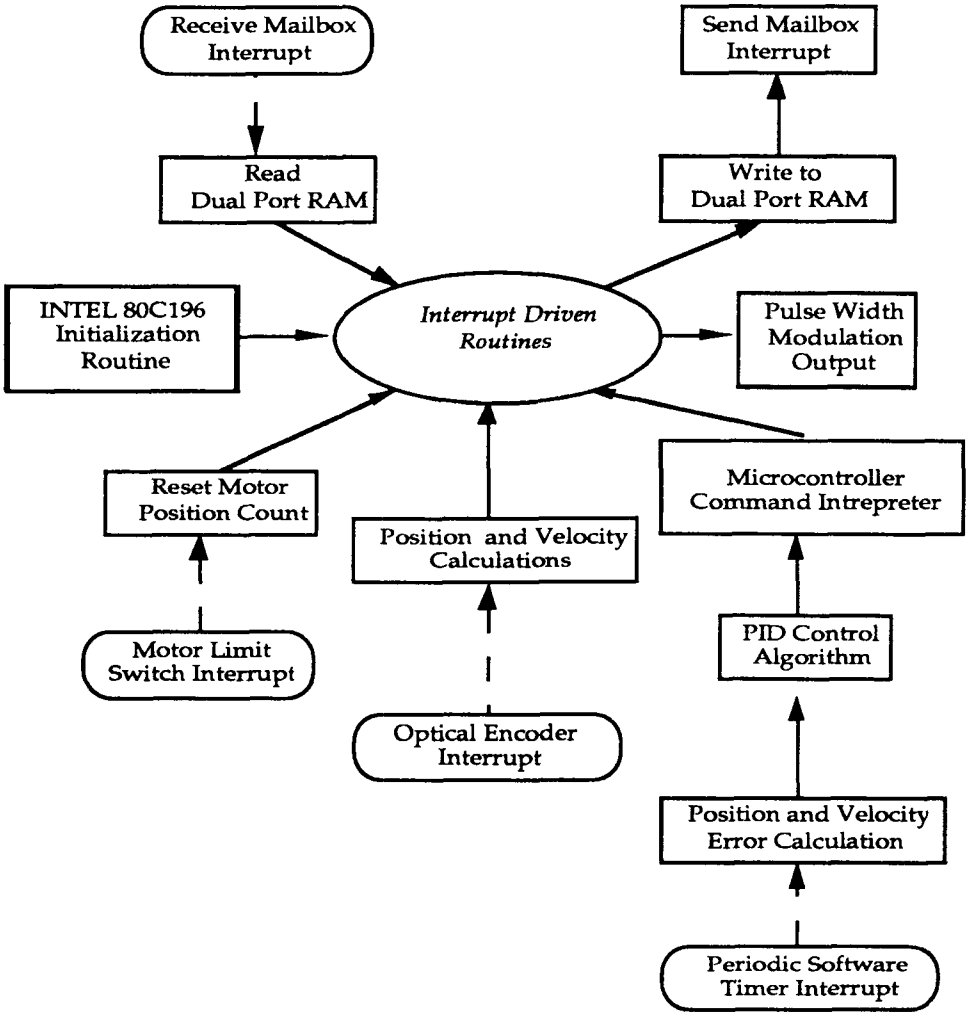

Receive Mailbox Interrupt
Send Mailbox Interrupt
Read Dual Port RAM
Write to Dual Port RAM
INTEL 80C196 Initialization Routine
Interrupt Driven Routines
Pulse Width Modulation Output
Reset Motor Position Count
Position and Velocity Calculations
Microcontroller Command Intrepreter
Motor Limit Switch Interrupt
Optical Encoder Interrupt
PID Control Algorithm
Position and Velocity Error Calculation
Periodic Software Timer Interrupt

Figure 4.3

Interprocessor Communication Protocol

The PC needs to be able to break down any possible action (for instance, a right turn to thirty degrees while walking at fifty-percent speed) into commands that are passed to the microcontrollers. In order to accomplish this, a PredaTerp Communication Protocol was established, consisting of a minimum set of commands combined with framework of parameters that can be written to the DP-RAMs. A table showing the seven minimum set commands is shown in Fig SW-4. The top axis shows the six parameters that are used as the communication protocol. An asterisk ("*") indicates that the command requires a valid value be passed in the corresponding parameter.

Command	Velocity	Height	Turn Position	Direction	Number of Steps
Walk	*	-	-	*	*
Change Velocity	*	-	-	-	-
Turn	*	-	*	-	-
YReset	-	-	-	-	-
XReset	-	-	-	-	-
Set Height	*	*	-	-	-
Stop	-	-	-	-	-

Figure 4.4

This table is straightforward. However, note that certain commands do not have some expected parameters. For instance, there is no direction specified for the "Turn" command. The human operator, or PredaTerp operating under an Autonomous Event Algorithm, may decide that it is necessary to turn "left" by thirty degrees. The proper parameters are sent by the PC's *Command Generator* to the *Write to Dual Port RAMs*, including the command to turn, the number of degrees the turn needs to be, and whether the turn is to the right or left.

It was decided to give all motors absolute positions, rather than attempt to define and work with relative positions. The microcontrollers reset their motors to a "zero position", and deal with only positive positional values. Therefore, the *Write to Dual Port RAM*s must convert the PC parameters into parameters that the microcontroller code will understand. The "Set Height" command works in a manner similar to the "Turn" command. The "Walk" command, however, requires that the "Direction" be sent to the microcontrollers. Since the walking motors go through complete revolutions, the absolute position values repeat themselves periodically. Therefore, in essence, we have to specify relative positions for walking.

The "Change Velocity" command is sent by the PC's *Write to Dual Port RAMs*, but is intercepted by the microcontrollers Read Dual Port RAM routine. This command causes a previous velocity parameter to be changed, without modifying the command (or any other) parameter. So, if the microcontroller was executing a "Walk" command at 50% velocity in the positive direction for 5 steps, a "Change Velocity" to 75% command would cause the microcontroller to execute a "Walk" command at 75% velocity in the positive direction for 5 steps.

Note that if an expected parameter field is not filled, the microcontroller coder will default to a set value. The "XReset", "YReset" and the "Stop" command require no parameters. In the case of the reset commands, the microcontroller proceeds to cycle its motors at a slow default speed until the reset switch is hit. The "Stop" command causes the microcontroller to cease all actions. Any motors in motion are brought to a stop at maximum deceleration.

Implementation of Interprocessor Communications

Several concerns arise when dealing with this communication scheme. In it present implementation, PredaTerp's Interprocessor Communication Protocol from PC to microcontroller is fairly simple. There are only seven commands, and up to five other parameters. Writing a complete command sequence to memory does not take a lot of time. However, consider that in order to walk, the PC needs to tell the six leg microcontrollers the same command. The time between when the first microcontroller gets its command and when the sixth microcontroller gets its command is a measurable delay. This timing delay could cause a problem as the legs could start out of phase. Consider the potential ramifications of an action that requires the six legs to walk at a constant speed, raise/lower their respective heights to different values to go over rough terrain, and turn the body, all simultaneously. This would create a temporal nightmare. Instead, consider Figure 4.5 below.

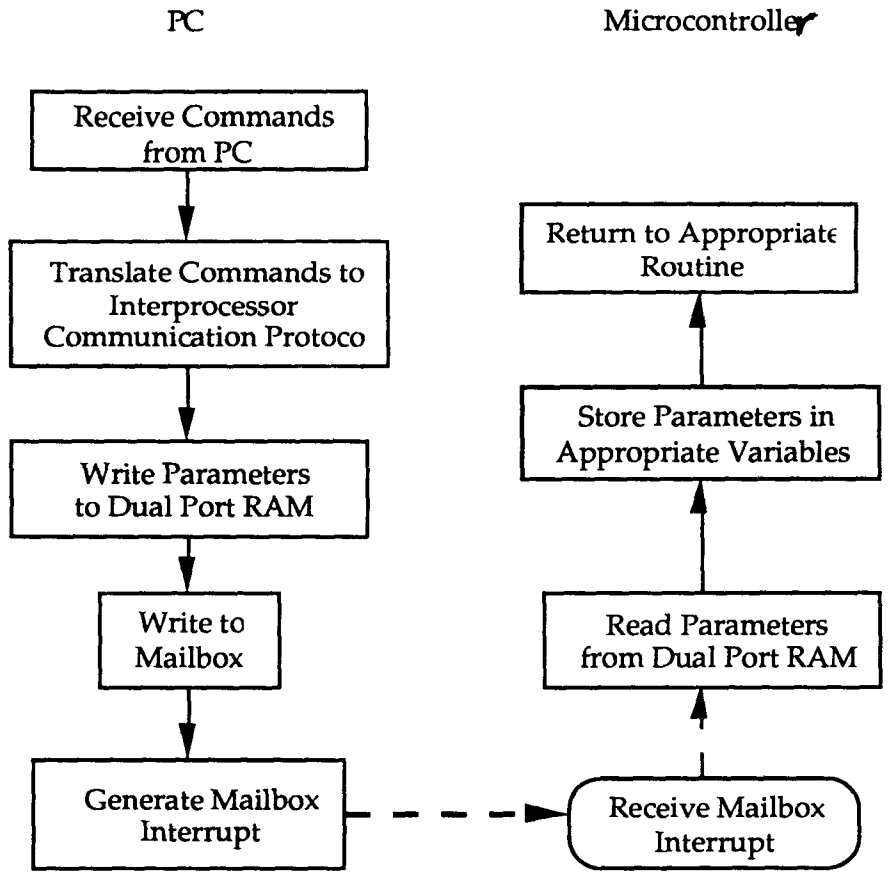


Figure 4.5

In order to minimize the timing problem, the PC writes commands to the appropriate microcontrollers' DP-RAM. The PC's Write to Dual Port RAMs routine then takes advantage of a DP-RAM's hardware feature. By writing a special one byte memory location on the DP-RAM (a "Mailbox"), an interrupt can be generated indicating that the DP-RAM contains information that needs to be read. The time between writing one byte on the first DP-RAM and one byte on the last DP-RAM is inconsequential.

There are two such memory locations per DP-RAM, so the microcontrollers can make use of this feature to let the PC know that there is feedback available (see Figure 4.6). When a microcontroller completes an expected action, it informs the PC by writing a message to the DP-RAM, and interrupting the PC. As all such interrupts from the DP-RAMs are ORed together and run to one input interrupt on the PC, when the PC receives the interrupt, it must poll the different lines to see which microcontroller wrote to its memory. While this system is not ideal, it does keep the PC from having to continually poll the various DP-RAMs to see if feedback from the microcontrollers are available. In addition, in an interrupt driven software system, the task of receiving feedback from the microcontrollers can be easily assigned relative importance in comparison to the other tasks that need performing at a given time.

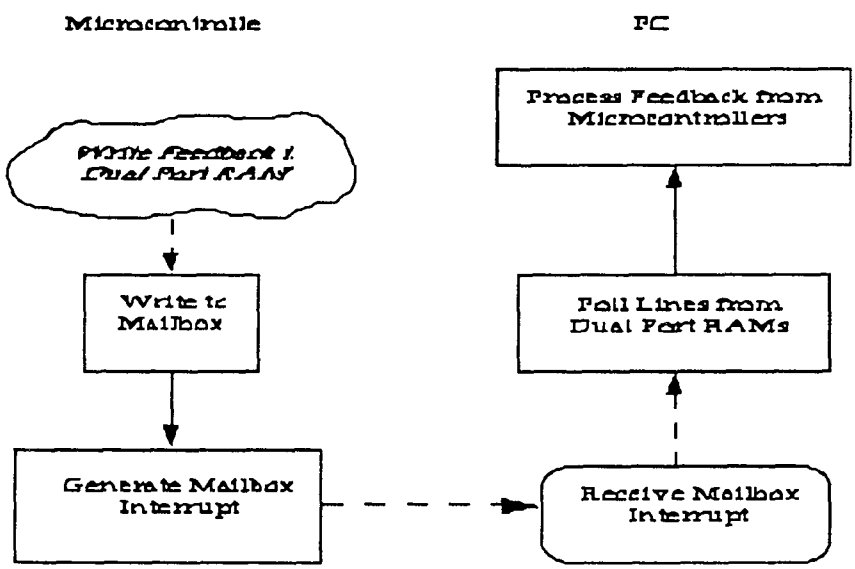


Figure 4.6

When the microcontroller *Receives Mailbox Interrupt,* it begins executing

the *Read Dual Port RAM* routine. This routine read the memory locations in which the parameters are written, and stores them in the appropriate variables for the *Microcontroller Command Interpreter* code to use. When the microcontroller notifies the PC when it completes a designated task by calling the *Write to Dual Port RAM* routine. Presently, this is the only feedback provided to the PC, so the *Write to Dual Port RAM* routine merely writes to the Mailbox. Future expansion of PredaTerp's capabilities may deem it necessary to provide more detailed feedback to the PC, but given the flexibility of the Dual Port RAM multiprocessor design, this is not seen to be a problem.

The Interprocessor Communication routines have one other function. The PC is a superb computational engine, with many fine mathematical libraries that are not readily available for the 80C196 family. Rather than try to compute a mathematically complex positional control equation on the microcontrollers in real time, it was decided that the PC should compute the desired motor positions once and place the values in a lookup tables stored in the seven DP-RAMs. The microcontrollers could access its DP-RAM just as if it were "standard" memory.

To accomplish the desired aims, the *Write to Dual Port RAMs* routine executes the calculation once, as the first step in the *System Initialization* routine. The values are stored in the appropriate memory locations, which the microcontrollers access in order to do the PID control calculations.

PC Software Implementation

The PC has four main tasks:

* Initialize itself and all supporting hardware,
* Process inputs,
* Make decisions regarding commands to be issued,
* Communicating with the various INTEL 80C196 microcontrollers.

Please refer back to Figure 4.2.

The *Main PC Initialization* routine is responsible for initializing the computing environment of the PC (i.e. creating variable memory locations, initializing constants, etc.). The PC is also responsible for coordinating the initialization of the various motors and encoders through the 80C196s (see Figure 4.7). PredaTerp cannot be sure that its initial state is known, or even that it is stable. That being the case, it could be disastrous if the microcontrollers started to initialize their corresponding motors without any guidance. The PC asks that the leg microcontrollers execute a "YReset". This causes all the legs to plant themselves, and raises the body to its full height. It also takes the height motors and encoders to a known position and resets the appropriate counters.

When the PC gets a "Command Done" from all the microcontrollers, it is ensured of a stable position. It then commands the microcontrollers controlling lower triangle's legs to raise their legs off the ground ("Set Height"). The PC then resets the corresponding walking motors/encoders by issuing an "XReset" command. Upon receiving the three "Command Done"s, the PC lowers the three legs attached to PredaTerp's lower triangle ("Set Height" ... "Command Done"), then raises the legs attached to the upper triangle ("Set Height" ... "Command Done"), and precedes in a similar fashion. Upon completion of the reset routine, the PC issues one more "Set Height" command to all the legs, placing PredaTerp at its operational height. The sequence of commands shown above reveals the power and flexibility of the Interprocessor Communication Protocol. By combining the available parameters in different orders, the PC is able to control the various motors with ease. In essence, the PC is executing commands in a simple robot control language. The commands issued by the *Autonomous Events Algorithms* and by the *Command Generator* follow a sequence not unlike that of the *System Initialization* routine.

The PC has the facility to process various inputs, ranging from the human tether interface to any number of possible environmental sensors. The standard PC keyboard was chosen to be the tether interface for PredaTerp. A custom tether could have been built, but it was unlikely that any custom tether could approach the flexibility afforded by the keyboard. Conceivably, a human operator could have in excess of 101 possible inputs, far more if multiple keystroke commands are used.

The human operator does not have control over individual functions of PredaTerp. For instance, the operator cannot drive a single motor directly, as could be had in a simple robot using a custom tether involving switches. In general, the human operator is constrained to command PredaTerp using variations of the Interprocessor Communication Protocol. For instance, an operator could direct the PC to command the turning motor's microcontroller to turn the motor thirty degrees to the left (Turn, Turning Motor, 30 degrees to left), but the operator could not switch that motor on directly. The reason for this is that, given the overall complexity of PredaTerp, no operator would be able to do any meaningful task in this manner. The ability to coordinate the six separate motors involved in a simple Walk command is beyond the capability of any operator.

Instead, it was felt that PredaTerp's PC code could be trusted to correctly decipher the operator directives and produce the correct microcontroller commands. This task involves writing relatively simple software (a directive parser, for instance) that is inherently robust and trustworthy. Given the fact that PredaTerp's tether is only operational when the PC is, it is safe to say that this is a valid assumption--it would be impossible to input human directives if the PC and corresponding code were rendered inoperable.

PredaTerp is also able to accept inputs via its voice command hardware. The software driving the voice command hardware operates the same way the keyboard tether does, only with much less flexibility. The human operator is constrained to five commands, as outlined in the Electrical Hardware section, under **Voice Recognition Circuitry**.

In addition to the human operator inputs, the PC is also able to easily accept inputs from any number of environmental sensors. The possible future additions of infrared sensors, tactile sensors, sonar, and possibly a full vision system would allow PredaTerp a flexibility in autonomous operation that it does not currently have.

Currently, the *Autonomous Events Algorithms* are constrained under limited environmental input. PredaTerp currently is currently configured to operate autonomously only in strictly defined, simple environments, such as the one pictured in Figure 4.7 below.

In such an environment, the desired trajectory can be computed ahead of time by PredaTerp's operators. The *Autonomous Event Algorithm* would be programmed to produce commands for the microcontrollers that would allow PredaTerp to navigate this environment with a minimum of environmental inputs.

The PC's last task is one that has already been discussed in the section titled **Interprocessor Communication Protocol.**

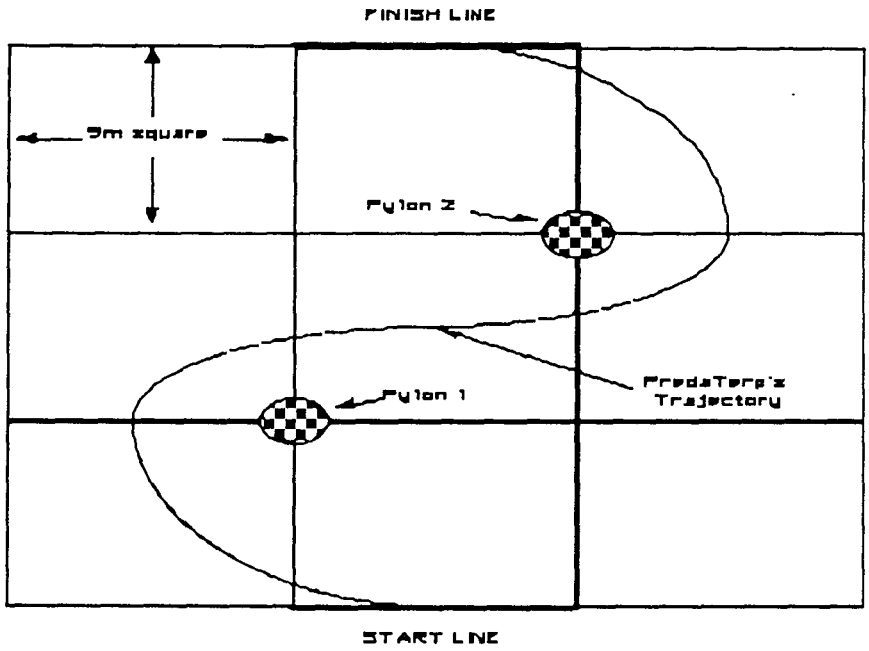


Figure 4.7

Microcontroller Software Implementation

It can be seen from Figure 4.3 that the microcontroller has six major functional tasks. The interprocessor communication routines have been discussed already, leaving the microcontroller initialization routine, the reset motor position routine, the feedback interpretation routine, the control algorithm and command interpreter routines, and the output to motors routine.

The initialization routine runs as soon as the microcontroller is powered up. It is primarily responsible for initializing constants and creating variable locations. Note that this is an initialization of the 80C196 computing environment only, not of the motors, encoders, or any other hardware. The initialization of these other systems is done under the guidance of the PC, via the reset commands ("XReset" and "YReset"). The INTEL 80C196 Initialization Routine is the only routine executing on the microcontrollers that is not initiated via an interrupt.

Control was instilled by way of an implementation of a Proportional, Differential, and Integral (PID) control algorithm. The PID routine is dependent on a constant period (dt) between its calculations. To ensure that the PID routine was executed at a set period, it was configured to run on every fifth Software Timer Interrupt.

$$\tau(t) = k_p e(t) + k_i \int e(t)dt + k_d(de/dt) \qquad \text{(Eq 4.1)}$$

The proportional term in the PID control equation shown in Eq 4.1 was easy to program. The integral term was implemented by keeping a sum of all previous errors, e(t). Periodically, the sum of errors was zeroed, so as to keep the value from growing to large to handle arithmetically. This is one of many accepted ways of handling the growing sum. The differential term was also simple in that all that was necessary was that a record be kept of the error that occurred just prior to the current time t. The three gains, k_p, k_i, k_d, are found by testing different values.

What units were to be used in the measure of dt? Consider that these are values based on the speed at which the 80C196 executes instructions. The measure of time is very small when compared to a second. It would take a lot of effort to create the floating point routines to do the calculations in 80C196 Assembly, at a great sacrifice of speed. In order to keep the calculations and Assembly routine simple, the interval of time dt was defined such that dt=1! This eliminates several tedious and slow floating point division and multiplication routines.

The *Optical Encoder Interrupt* records the encoder position values as they come in. The PID routine calculates error based on the encoder position value used in its last calculation, ΔP_{last}, and the most recent encoder value that was recorded, $\Delta P_{present}$. The values that fall in between ΔP_{last} and $\Delta P_{present}$ are extraneous, as they fall within the time period dt, not on the boundaries. Once the PID calculation is done, an other routine converts $\tau(t)$ to the appropriate *Pulse Width Modulation Output* value that can be used to drive the motors.

$$\Delta P = P_{desired} - P_{actual} \qquad \text{(Eq 4.2)}$$

It is important to note that the PID routine calculates errors based on actual motor position, which is supplied by the optical encoders, and the expected motor position, which is calculated before the PID routine is called. When is this done? The answer lies in Figure 4.3. Notice that the *PID Control Algorithm* is followed by the *Microcontroller Command Interpreter*? The *Microcontroller Command Interpreter* is responsible for taking the information that the PC sent and calculating the next desired position for the motor(s). So, the next desired position, P_{next}, is found right after the PID routine calculates the value necessary to move the motor to $P_{desired}$ by the next dt interval.

The last microcontroller routine is that of the *Motor Limit Switch Interrupt*. The Reset Motor Position Counter for the x-trajectory (walking) is enabled only when the microcontroller is issued a reset command by the PC. This allows the microcontroller to zero the motor position counter at a known position in the leg's trajectory. In normal operation, the walking limit switch is ignored, as it has a complete range of motion that it can go through. The body limit switches and the height limit switches are not ignored during their operation, as there are definite limits as to how far a leg can extend/retract or a body turn.

Future PredaTerp Capabilities

PredaTerp currently employs the bare minimum hardware and software necessary to perform simple tasks. The employment of a IBM PC clone "core" computing engine allows for easy expansion of PredaTerp's capabilities, ranging from sophisticated sensors, to image processing, expanded voice recognition, voice synthesis, and "artificial intelligence". The addition of mechanical actuators would enable PredaTerp to manipulate its environment, not just navigate in it.

On a smaller scale, PredaTerp could use power consumption monitoring capabilities, so as to conserve its batteries by shutting down non-vital, power draining components. Expanding the memory available to the 80C196s would allow for more sophisticated control algorithms to be implemented.

The PredaTerp designers have succeeded in developing a versatile and functional walking machine. In its next iteration, PredaTerp will be truly worthy of the title "Walking Robot".

Conclusion

The design and manufacture of a walking machine was completed by thirty students in seven months. The responsibilities were divided into leg, body, electrical hardware and software tasks. The mechanical and electrical engineering students were instructed over two semesters through the design and construction processes.

The leg design combined a modified crank and rocker mechanism with pantograph and leg lift mechanism. The six legs each operate with two degrees of freedom, providing great flexibility. Structural integrity was maintained through computer engineering analysis and numerical control machinery.

The body design provided a third degree of freedom for the robot. This was achieved with a turning mechanism. This mechanism controls the relative position of the two body frames. The rigid tripod frames serve as a means to mount the six legs and the electrical hardware components.

The electrical hardware design employed distributed processing and modular components to control and power the walking machine. A supervisory computer accepts commands, oversees control and runs autonomous programs. Microprocessors were used to directly control the thirteen motors. Communication between the PC and microprocessors is performed with dual port RAM.

The software design coordinated the robots actions. Low level code written to the microcontrollers controls the motor positions. High level code written to the PC processes programs and commands. Communications code breaks down PC commands into smaller microcontroller tasks and coordinates timing of data. The robot presently employs the bare minimum hardware and software necessary to perform simple tasks. The use of a IBM PC clone "core" computing engine allows for easy expansion of robot's capabilities, ranging from sophisticated sensors, to image processing, expanded voice recognition, voice synthesis, and "artificial intelligence".

Practical applications were also considered in the walking robot's design. The machine is easily adaptive to almost any terrain due to the design's flexibility. In addition, the mechanical actuators would enable the robot to manipulate its environment, not just navigate in it. The leg design emulates a human stride, allowing a modified system to serve in functions hazardous to humans. The

feedback control design allows the robot to be adapted to perform repeatable, precise tasks.

The University of Maryland robot designers have succeeded in developing a versatile, multi-functional walking machine. With adjustments to the basic design, the capabilities of the robot can be directed to many applications – whether they be simple and close to home, or complex and as far away as the face of the moon.

APPENDIX:

Leg Motors and Link Dimensions

PITMO® D-C SERVO MOTORS

Series 14000 - 2.125 in. O.D.
with Stall Torques from 160 to 286 oz.-in.

This family of permanent magnet field motors offers significantly higher performance than the Pittman® 13000 series through the use of an 11-slot armature lamination designed to use most advantageously the high air gap flux densities provided by radially oriented Ceramic 8 magnets. Series 14000 servo motors have been developed, produced and proved for long, maintenance-free operation. Premium quality materials coupled with the very latest manufacturing and assembly techniques provide excellent reliability. In addition, every motor is subjected to complete testing of all critical parameters under both load and no load conditions in the unique Pittman® computerized final testing station. A printout of test data is kept on file for any further reference.

Speed, voltage, current and torque characteristics can be varied over a wide range to meet specific needs. Please note that armature winding changes, and any relatively simple modifications that do not require extensive redesign or tooling alterations, may be specified for prototype quantities at only nominal costs.

PRIMARY DESIGN FEATURES OF THE SERIES 14000

PEAK TORQUE (STALL)
from 160 to 286 oz.-in.

NO LOAD SPEEDS
from about 3,000 to 3,700 rpm for standard motors at rated voltages

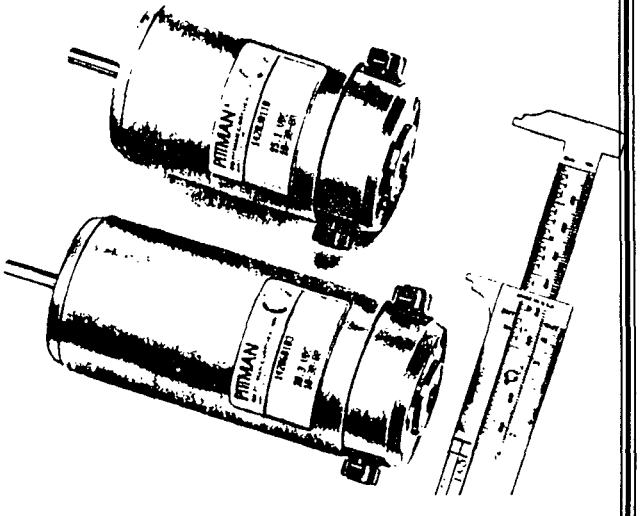
ARMATURES
11-slot design, skewed for reduction of reluctance torque. Laminations are silicon steel, with standard windings of film-insulated (class 200°C) magnet wire — impregnated with polyester resin and baked.

COMMUTATORS
diamond turned after armature assembly to ensure optimum concentricity and long brush life.

BRUSHES
copper-graphite standard.
Optional materials at additional costs include silver-graphite and other specified material compositions.

FIELD
Radially oriented strontium-ferrite magnets enclosed in heavy-gage steel return rings. End bells are zinc die castings.

BEARINGS
self-aligning sintered bronze, precisely sized to provide optimum journal clearance. Also equipped with felt wicks for reserve lubrication. Optional double shielded ball bearings available at additional cost.



PITMAN®
HARLEYSVILLE, PA 19438-0003 USA
A DIVISION OF PITTMAN ENGINEERING & MANUFACTURING CORP

TELEPHONE (215) 256-6601
FAX (215) 256-1338

TELEX 283668
TWX 510-661-8696

Figure A-1

MOTOR SIZE CONSTANTS

ITEM	MOTOR SIZE CONSTANTS	UNITS	SYMBOL	14202 VALUE	14203 VALUE	14204 VALUE	14205 VALUE	14206 VALUE
1	PEAK TORQUE (STALL)	OZ-IN	TPK	106	160	205	226	286
1 a	Peak torque (stall)	N•m	TPK	0.75	1.13	1.45	1.60	2.02
2	MOTOR CONSTANT	OZ-IN/$\sqrt{W}$	PKO	5.81	7.88	8.63	9.97	10.9
2 a	Motor constant	mN•m/$\sqrt{W}$	PKO	41.1	55.6	60.9	70.4	77.0
3	POWER FOR PEAK TORQUE	W	PWR	333	417	570	519	686
4	DAMPING (ZERO SOURCE IMPED.)	OZ-IN/(rad/s)	DPO	0.248	0.439	0.526	0.701	0.841
4 a	Damping (zero source imped.)	mN•m/(rad/s)	DPO	1.75	3.10	3.71	4.95	5.94
5	DAMPING (INFINITE SOURCE IMPED.)	OZ-IN/(rad/s)	DPI	7.4×10^{-4}	8.5×10^{-4}	10.3×10^{-4}	11.2×10^{-4}	13.0×10^{-4}
5 a	Damping (infinite source imped.)	mN•m/(rad/s)	DPI	5.2×10^{-3}	6.0×10^{-3}	7.3×10^{-3}	7.9×10^{-3}	9.2×10^{-3}
6	NO LOAD SPEED	REV/MIN	SNL	3820	3420	3670	3040	3170
6 a	No load speed	rad/s	WNL	400	358	384	318	332
7	ELECTRICAL TIME CONSTANT	ms	TCE	1.47	1.64	1.58	1.63	1.62
8	MECHANICAL TIME CONSTANT	ms	TCM	8.5	6.8	7.0	6.3	6.2
9	FRICTION TORQUE	OZ-IN	TOF	2.0	2.0	2.0	2.0	2.0
9 a	Friction torque	mN•m	TOF	14.1	14.1	14.1	14.1	14.1
10	ARMATURE INERTIA	OZ-IN-s^2	ERT	2.3×10^{-3}	3.0×10^{-3}	3.7×10^{-3}	4.4×10^{-3}	5.2×10^{-3}
10 a	Armature inertia	kg•m^2	ERT	16.3×10^{-6}	21.2×10^{-6}	26.1×10^{-6}	31.1×10^{-6}	36.7×10^{-6}
11	MOTOR WEIGHT	OZ	WGT	26.0	31.2	35.2	39.5	45.4
11 a	Motor mass	kg	WGT	0.74	0.88	1.0	1.1	1.3
12	THEORETICAL ACCELERATION	rad/s^2	CEL	46100	53300	55400	51300	55000
13	THERMAL TIME CONSTANT	MIN	TCT	24.0	26.0	28.8	29.4	33.6
14	ULTIMATE TEMP. RISE/WATT	°C/W	TPR	9.0	8.1	7.7	7.3	6.8
15	MAXIMUM WINDING TEMPERATURE	°C	TMX	155	155	155	155	155

WINDING CONSTANTS (other windings available)

MODEL 14202

*	WINDING CONSTANTS	UNITS	SYMBOL	WDG #1	WDG #2	WDG #3	WDG #4
16	VOLTAGE	V	VLT	12.0	19.1	24.0	30.3
17	CURRENT (STALL)	A	AMP	27.8	17.6	13.8	11.0
18	TORQUE CONSTANT	OZ-IN/A	TPA	3.90	6.15	7.80	9.85
18 a	Torque constant	mN•m/A	TPA	27.5	43.5	55.1	69.6
19	TERMINAL RESISTANCE	OHMS	RTR	0.431	1.08	1.73	2.76
20	BACK EMF	V/(rad/s)	BEF	0.028	0.043	0.055	0.070
21	INDUCTANCE	mH	DUK	0.635	1.58	2.54	4.05
22	CURRENT (NO LOAD)	A	INL	0.460	0.291	0.230	0.182

MODEL 14203

*	WINDING CONSTANTS	UNITS	SYMBOL	WDG #1	WDG #2	WDG #3	WDG #4
16	VOLTAGE	V	VLT	12.0	19.1	24.0	30.3
17	CURRENT (STALL)	A	AMP	34.8	21.8	17.4	13.7
18	TORQUE CONSTANT	OZ-IN/A	TPA	4.63	7.41	9.26	11.7
18 a	Torque constant	mN•m/A	TPA	32.7	52.3	65.4	82.8
19	TERMINAL RESISTANCE	OHMS	RTR	0.345	0.877	1.38	2.21
20	BACK EMF	V/(rad/s)	BEF	0.033	0.052	0.065	0.083
21	INDUCTANCE	mH	DUK	0.565	1.45	2.26	3.63
22	CURRENT (NO LOAD)	A	INL	0.390	0.244	0.195	0.154

MODEL 14204

*	WINDING CONSTANTS	UNITS	SYMBOL	WDG #1	WDG #2	WDG #3	WDG #4
16	VOLTAGE	V	VLT	12.0	19.1	24.0	30.3
17	CURRENT (STALL)	A	AMP	47.7	30.2	23.8	19.1
18	TORQUE CONSTANT	OZ-IN/A	TPA	4.34	6.86	8.67	10.8
18 a	Torque constant	mN•m/A	TPA	30.6	48.5	61.2	76.5
19	TERMINAL RESISTANCE	OHMS	RTR	0.251	0.633	1.01	1.59
20	BACK EMF	V/(rad/s)	BEF	0.031	0.048	0.061	0.076
21	INDUCTANCE	mH	DUK	0.400	1.00	1.60	2.50
22	CURRENT (NO LOAD)	A	INL	0.420	0.265	0.210	0.168

* OTHER WINDINGS AVAILABLE

Figure A-2

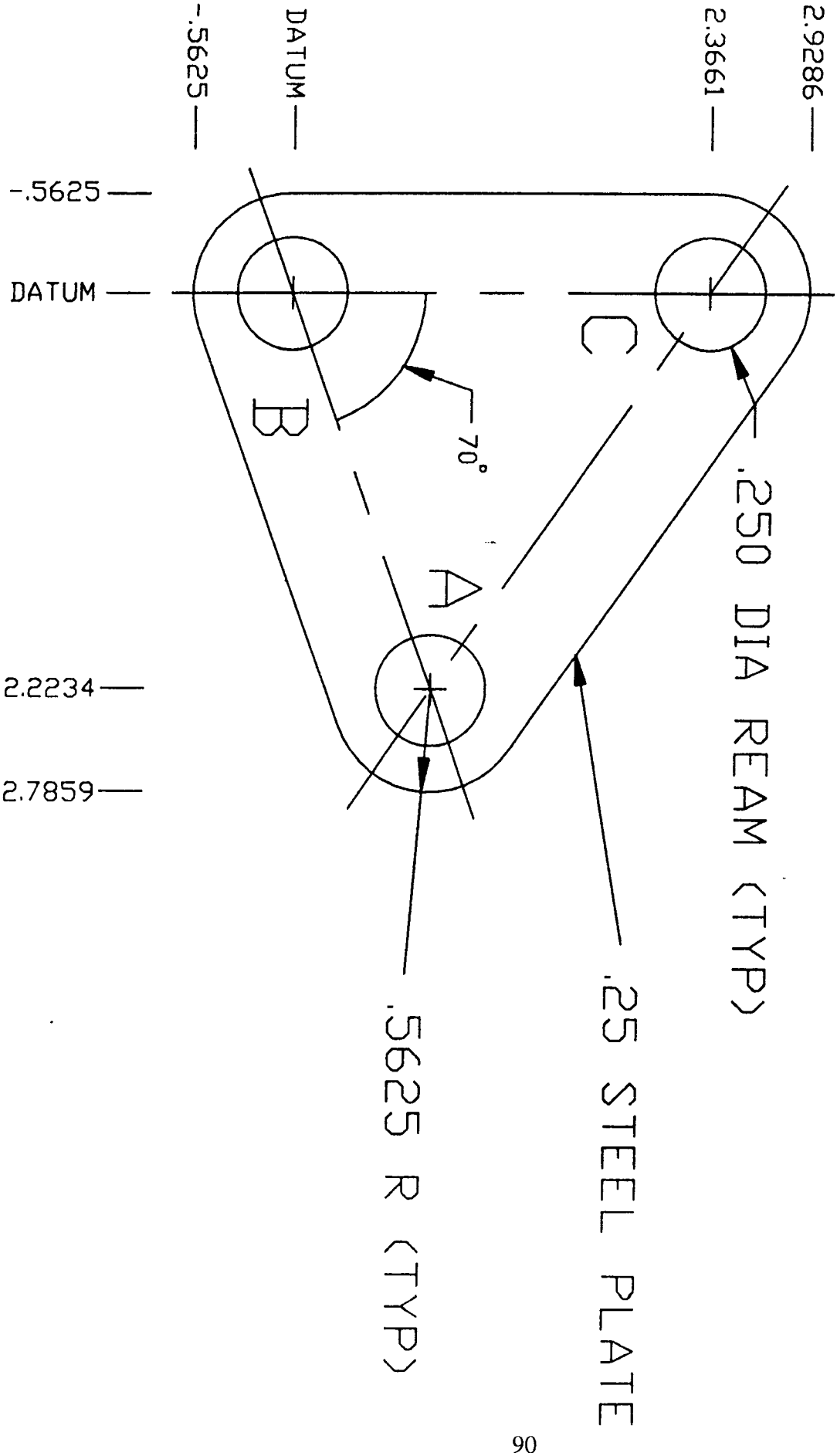

-.5625
DATUM
2.3661
2.9286
-.5625
DATUM
2.2234
2.7859
70°
A
B
C
.250 DIA REAM (TYP)
.25 STEEL PLATE
.5625 R (TYP)

Figure A-3

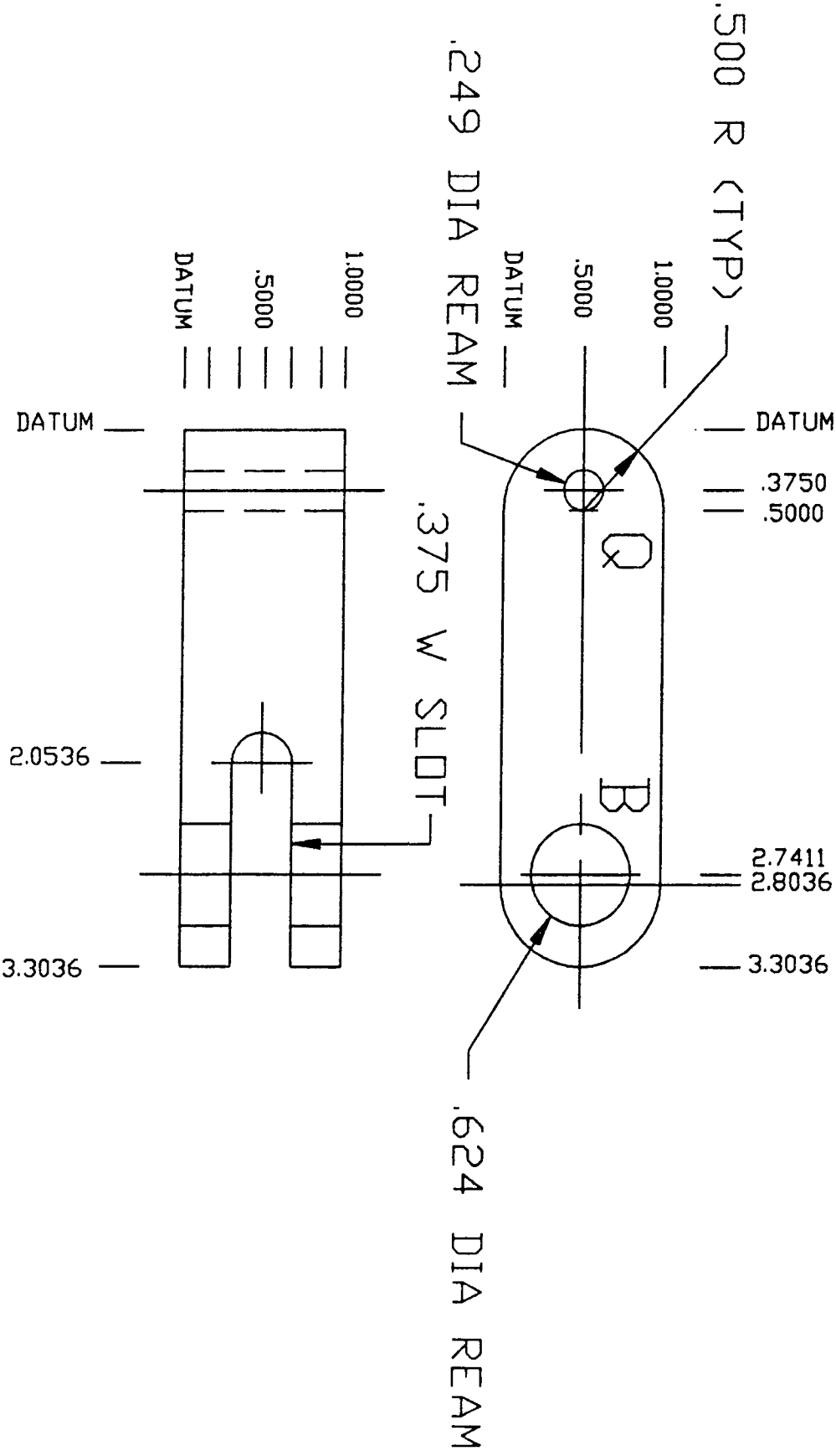

.500 R (TYP)
.249 DIA REAM
.375 W SLOT
.624 DIA REAM
DATUM
.5000
1.0000
DATUM
.3750
.5000
2.0536
2.7411
2.8036
3.3036
3.3036
Q
B

Figure A-4

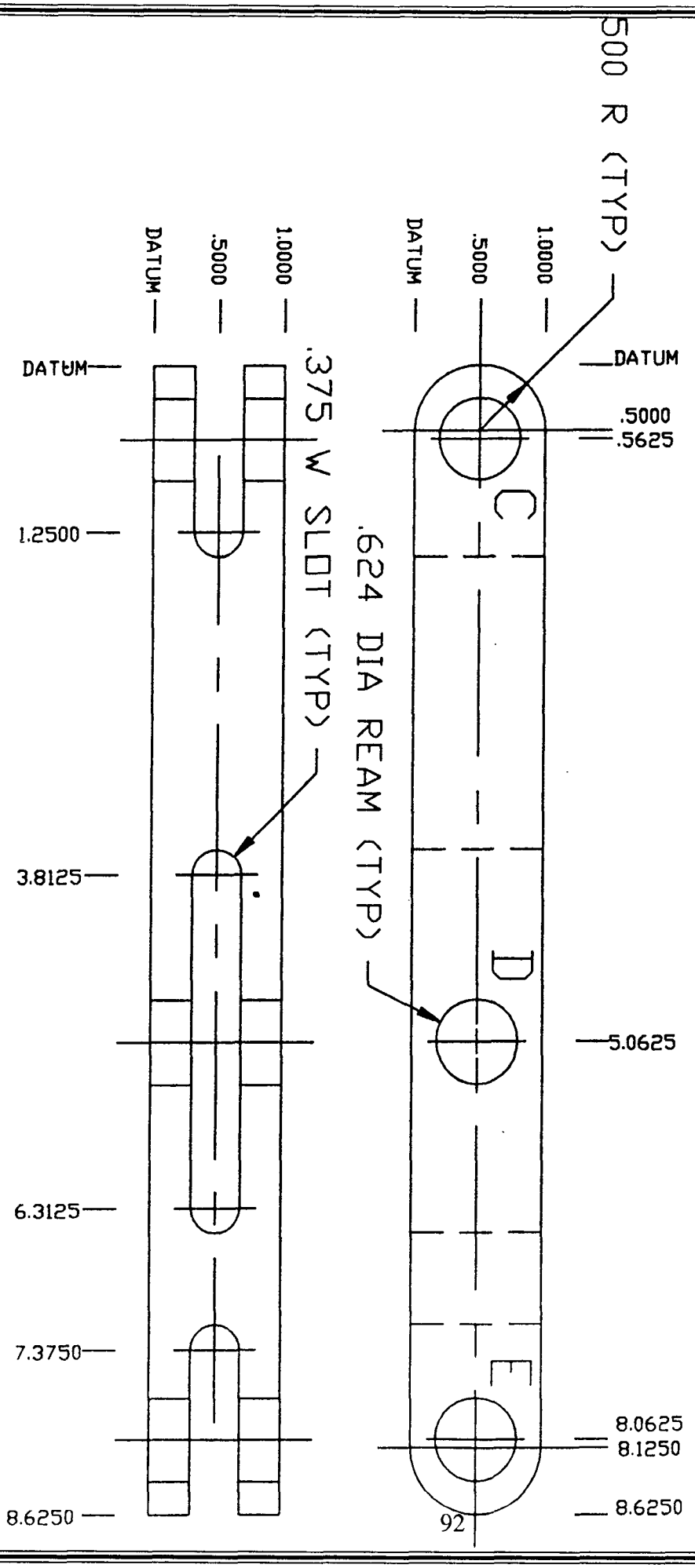

.500 R (TYP)
DATUM
.5000
1.0000
DATUM
.5000
.5625
C
D
E
.624 DIA REAM (TYP)
5.0625
8.0625
8.1250
8.6250
.375 W SLOT (TYP)
DATUM
1.2500
3.8125
6.3125
7.3750
8.6250

Figure A-5

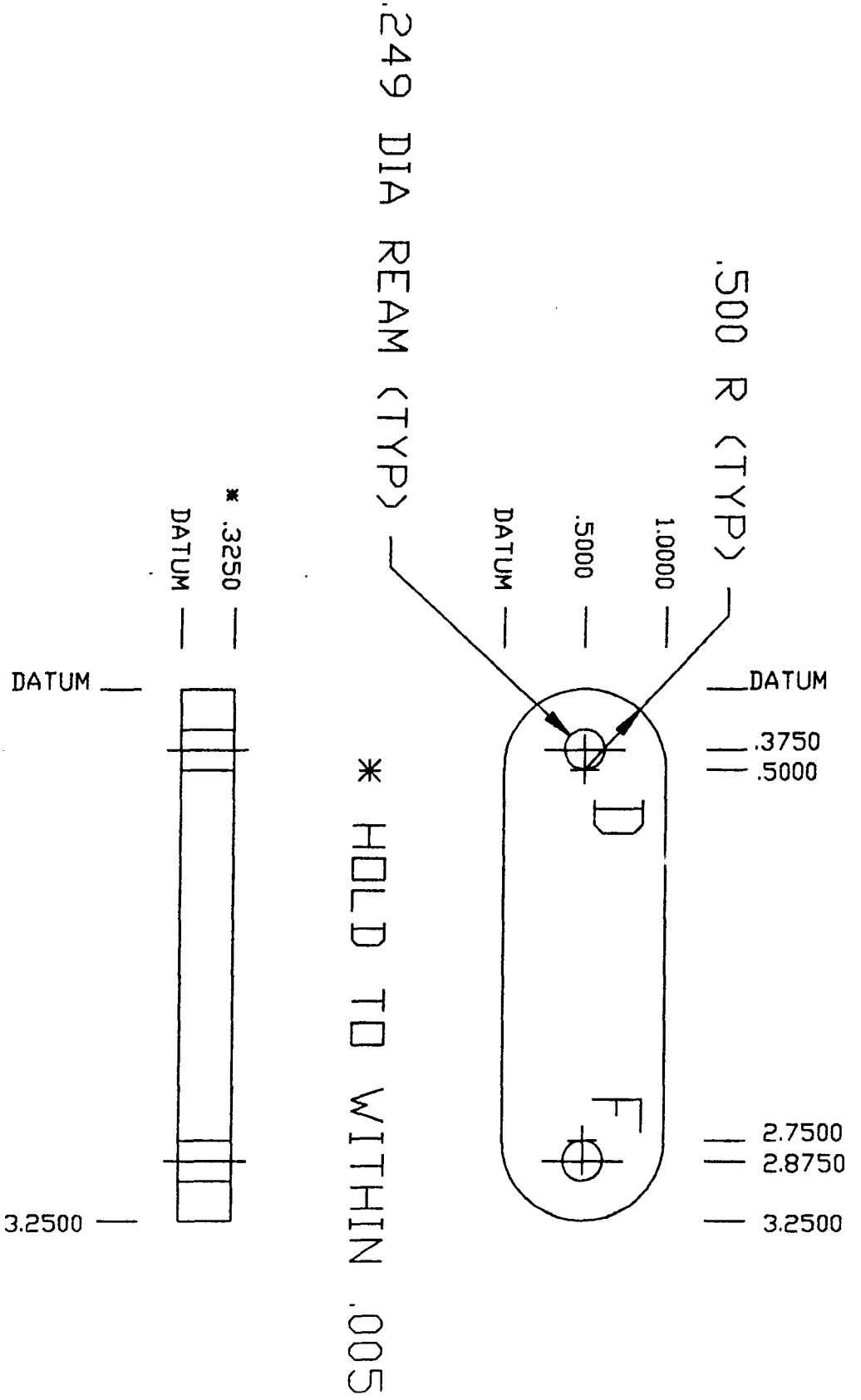

.249 DIA REAM (TYP)
.500 R (TYP)
* HOLD TO WITHIN .005
DATUM
* .3250
DATUM
DATUM
3.2500
DATUM
.5000
1.0000
DATUM
.3750
.5000
2.7500
2.8750
3.2500
D
F

Figure A-6

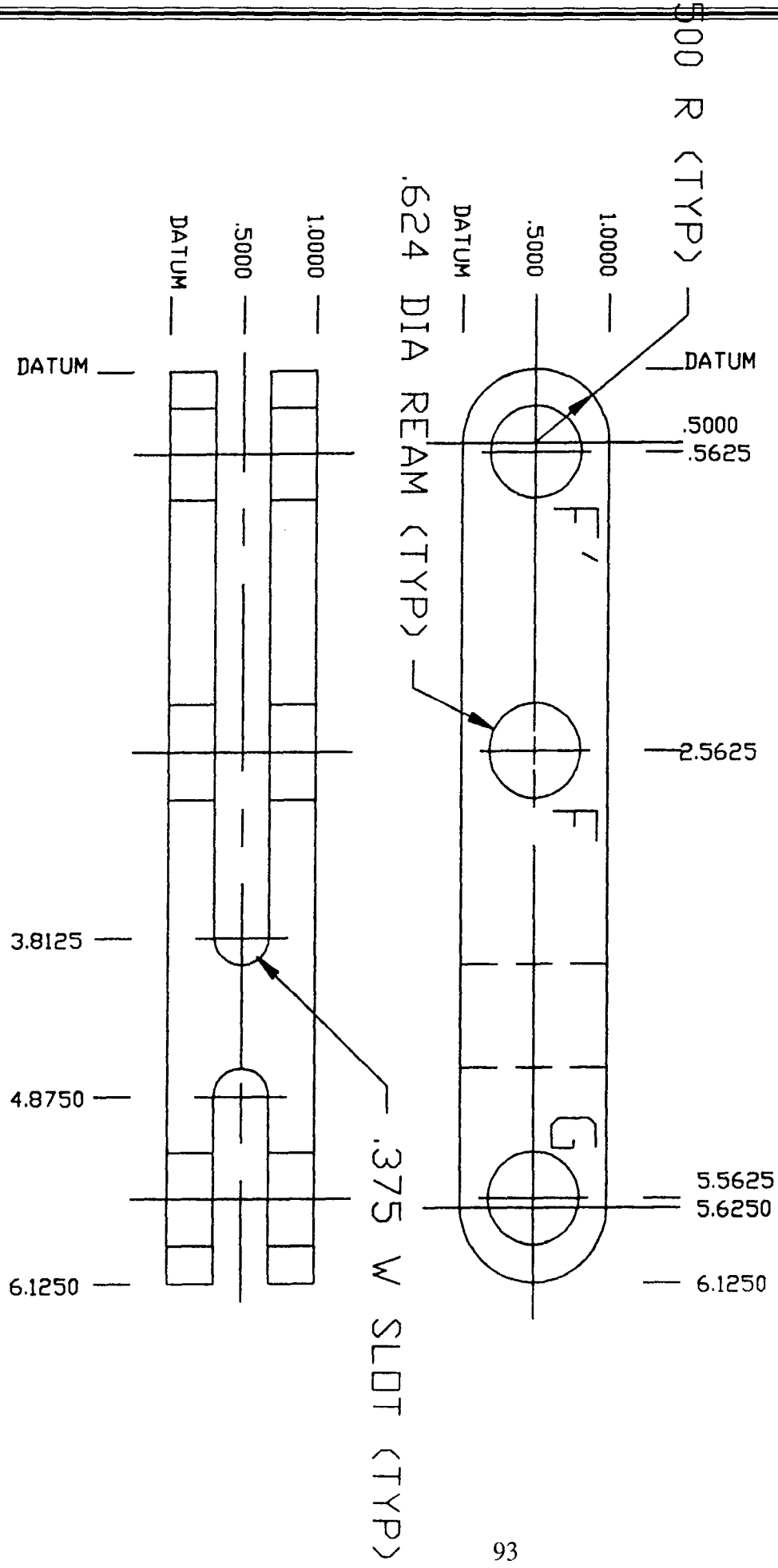

.500 R (TYP)
DATUM
.5000
1.0000
DATUM
.624 DIA REAM (TYP)
.5000
.5625
2.5625
5.5625
5.6250
6.1250
3.8125
4.8750
6.1250
.375 W SLOT (TYP)
F'
F
G

Figure A-7

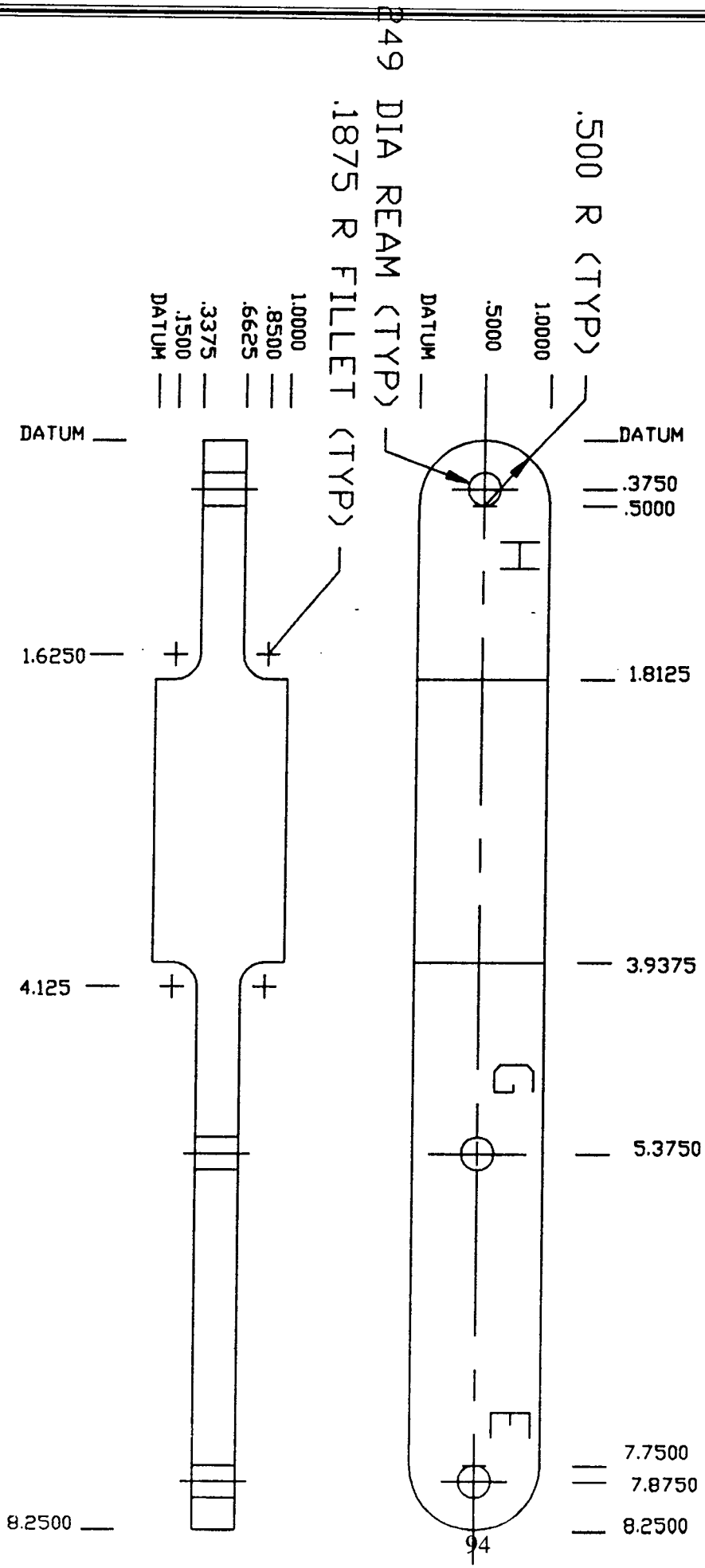

.500 R (TYP)
.49 DIA REAM (TYP)
.1875 R FILLET (TYP)
DATUM
1.0000
.8500
.6625
.3375
.1500
DATUM
1.6250
4.125
8.2500
DATUM
.5000
1.0000
DATUM
.3750
.5000
1.8125
3.9375
5.3750
7.7500
7.8750
8.2500
H
G
E

Figure A-8

References

Electrical Hardware Group

Archer, VCP200 Application Notes, Archer, Fort Worth: 1988.

Cooper, Daniel B.; "Experiments in Voice Recognition," Radio Electronics: April 1991.

"87C196KB 16-Bit High Performance CHMOS Microcontroller," 16-Bit Embedded Controllers, Intel Corporation:1990.

"L296 Switching Power Supply", Linear Integrated Circuits DATABOOK, Unitrode Corporation, Merrimack NH: 1987.

Schaefer, Tim; Michael Chevalier, Distributed Motor Control Using the 80C196KB, Intel Corporation: May 1989.

Snow, C. Bruce; "Making PC Boards," Radio Electronics: November 1987.Intel 16-bit Embedded controllers (1996), Inlet Corporation, Order Number 270646.

Software Group

Intel Microprocessor and Peripheral Handbook, vol. 1 & vol. 2 (1989), Inlet Corporation, Order Number 230843.

Inlet 8086 / 8088 Users Manual (1989), Inlet Corporation, Order Number 240487-001.

Intel Embedded Control Applications Handbook (1989), Inlet Corporation, Order Number 270648.

Schilling, Robert J.,(1990), Fundamental Of Robotics; Analysis of Control , ISBN 0-13-344433-3.

Shade, Gray, (1985), 8088 IBM PC Assembly Language Programming , ISBN 0-03-001298-8.

Hancock, Les, & Krueger, Morris (1982), The C Primer , ISBN 0-07-02598.

Thorne, Michael (1986), Programming The 8086/8088 , ISBN 0-8053-5004-7.